



**KING KHALED UNIVERSITY**  
**COLLAGE OF ART AND SCIENCE IN**  
**KHAMIS MUSHAYT**  
**INFORMATION SYSTEM DEPARTEMENT**  
**BILLS ANALYSIS SYSTEM**

**Prepared By:**

**Farah**  
**Reuof**  
**Lubna**  
**Lamees**  
**Amani**  
**Nada**  
**Shahad**

**Supervised by:**

**Dr. Basmah Almoaber**  
**Assistant professor**

**This Project Documentation is submitted in partial fulfillment of the  
requirement for the Degree of B.Sc. in Information Systems**

**Academic Year: 2022/2023**

## COMMITTEE REPORT

We certify that we have read this graduation venture report as looking at advisory group, analyzed the understudy in the substance and that as we would like to think it sufficient as an undertaking archive for B.Sc. in Information Systems

**Supervisor:**

Name: Dr. Basmah  
Almoaber

Signature . . . . .

Date:

**Examiner1:**

Name: .....

Signature . . . . .

Date:

**Examiner2:**

Name: .....

Signature . . . . .

Date:

## **ACKNOWLEDGEMENT**

First of all we would like to express our gratitude to Almighty God for enabling us To completethis project on " Bill's Analysis System".

Apart from our efforts, the success of the project depends to a large extent on the encouragement and guiding principles of many others. We now take this opportunity to express our gratitude to the people who had an instrumental role in the success of this project.

We would like to express our deepest gratitude to the supervisor and Head of Information Systems Department Prof. Basmah Almoaber for her kind support to complete the project. And directing and supervising everything related to the project.

We would also like to thank our families for their support and standing by us from the beginning of work on this project until its success.

## **ABSTRACT**

At the present time, expenses are increasing day by day putting more weight on families' shoulders. In many cases, monthly bill expenses may increase more than usual, and families fail to notice. One of the reasons for noticing bill changes is that, families are paying each bill separated on the website of the bills issuer. To help families manage their expenses and pay their bills efficiently, we are planning to create a website that analyzes the bills and alerts families in the event of any significant increase or decrease in bill expenses.

The data analysis part of the system will be built using the Python programming language and linked to a website, as we seek to benefit from technological development and spread it in the Kingdom of Saudi Arabia

**Keyword:** *Data Analysis, Bills, monthly consumption.*

## **TABLE OF CONTENT**

<b>COMMITTEE REPORT .....</b>	<b>2</b>
<b>ACKNOLEDGEMENT.....</b>	<b>3</b>
<b>ABSTRACT .....</b>	<b>4</b>
<b>LIST OF TABLES .....</b>	<b>8</b>
<b>LIST OF FIGURES .....</b>	<b>9</b>
<b>LIST OF ABBREVIATIONS .....</b>	<b>10</b>
<b>1. CHAPTER 1: INTRODUICION.....</b>	<b>11</b>
1.1. INTRODUCTION .....	11
1.2. PROBLEM STATEMENT .....	11
1.3. PROBLEM SOLUTION .....	11
1.4 AIMS OF THE PROJECT .....	12
1.5. OBJECTIVES OF THE PROJECT .....	12
1.6. SCOPE OF THE PROJECT.....	12
1.7. SIGNIFICANCE OF THE PROJECT (MOTIVATION).....	12
1.8. SOFTWARE DEVELOPMENT METHODOLOGIES .....	12
1.9. PROJECT ORGANIZATION.....	14
1.10. SUMMARY .....	15
<b>2. CHAPTER 2: BACKGROUND AND LITERATURE REVIEW .....</b>	<b>16</b>
2.1. LITERATURE REVIEW.....	16
1. AlKahraba app.....	16
2 MINT App .....	17
3. Phone bill Analyzer App.....	18
2.2. COMPARISON TABLE BETWEEN EXISTING SYSTEMS AND PROPOSED SYSTEM ..	19
2.3. FEASIBILITY ANALYSIS .....	20
2.3.1. TECHNICAL FEASIBILITY .....	20
2.3.2. ECONOMICAL FEASIBILITY .....	21
2.4. REQUIREMENTS ANALYSIS .....	21
2.4.1. FUNCTIONAL REQUIREMENT .....	21
2.4.2. NON-FUNCTIONAL REQUIREMENT.....	22
2.5. SUMMARY.....	23
<b>3. CHAPTER 3 SYSTEM ANALYSIS AND SYSTEM DESIGN .....</b>	<b>24</b>

3.1.	INTRODUCION .....	24
3.2.	INTERACTION ANALYSIS .....	24
3.2.1.	UML DIAGRAM .....	24
3.3.	DESIGN CONCEPT .....	31
3.4.	CONTEXT FLOW DIAGRAMS .....	32
3.5.	DATA FLOW DIAGRAM .....	32
3.6.	DATABASE DESIGN .....	33
3.6.1.	ER DIAGRAM .....	33
3.6.2.	DATABASE TABLES .....	35
3.7.	SAMPLE INTERFACE DESIGN .....	38
3.8.	UNDERLYING TEQNIQUES .....	44
3.9	SUMMARY .....	44
4.	CHAPTER 4 SYSTEM IMPLEMENTATION AND RESULTS .....	45
4.1.	INTRODUCTION .....	45
4.2.	INTERFACE DESIGN AND CODING .....	45
4.2.1.	BILLS DATASET .....	45
4.2.2.	LINEAR REGRESSION MODELS .....	52
4.2.3.	LANDING PAGE MODULE .....	64
4.2.4.	DASHBOARD WEB APPLICATION DEVELOPMENT .....	67
4.2.5.	MODEL CLASSES MODULE .....	68
4.2.6.	CONTROLLER CLASSES MODULE .....	70
4.2.7.	REGISTER MODULE .....	71
4.2.8.	LOGIN MODULE .....	73
4.2.9.	ADD BILL MODULE .....	75
4.2.10.	VIEW BILL MODULE .....	77
4.2.11.	VIEW BILL MODULE .....	79
4.2.12.	DASHBOARD home-page module .....	81
4.2.13.	BUDGET module .....	82
4.2.14.	REPORT MODULE .....	84
4.2.15.	NOTIFICATIONS MODULE .....	85
5.	CHAPTER 5 TESTING AND EVALUATION .....	87
5.1.	INTRODUCTION .....	87

5.2.	TEST PLAN AND OBJECTIVES .....	87
5.3.	STAGES OF TESTING .....	88
5.3.1.	WHITE BOX TESTING.....	88
5.3.2.	BLACK BOX TESTING .....	92
5.4.	TEST CASES AND TEST RESULTS.....	94
5.4.1.	USER LOGIN.....	94
5.4.2.	ADD BILL .....	94
5.4.3.	LINEAR REGRESSION RESULTS .....	95
5.5.	SUMMARY .....	96
6.	CHAPTER 6 CONCLUSION AND FUTURE WORK.....	97
6.1.	CONCLUSION.....	97
6.2.	LIMITATIONS OF THE PROJECT.....	97
6.3.	FUTURE ENHANCEMENTS.....	97
	REFERENCES .....	98

## LIST OF TABLES

Table 1.1: Member's Task .....	14
Table 2.1: Comparison Table with existing systems.....	19
Table 4.1: Datasets independent variables .....	46
Table 4.2: Electricity dataset Gaussian distribution .....	46
Table 4.3: Phone dataset Gaussian distributions .....	48
Table 4.4: Water dataset Gaussian distributions .....	49
Table 4.5: Gas dataset Gaussian distributions .....	51
Table 4.6: Add bill result code.....	71
Table 5.1: Register module unit testing .....	89
Table 5.2: Login module unit testing.....	89
Table 5.3: AI module units testing .....	89
Table 5.4: User authentication and dashboard integration .....	91
Table 5.5: Add bill and AI modules integration .....	92
Table 5.6: AI modules and result modules integration .....	92
Table 5.7: System testing results .....	93
Table 5.8: Acceptance testing result.....	93
Table 5.9: User login test case .....	94
Table 5.10: Add bill test case .....	94



## LIST OF FIGURES

Figure 1.1: Agile methodology.....	13
Figure 1.2: Gantt Chart .....	15
Figure 2.1: Alkahraba app .....	17
Figure 2.2: MINT app .....	18
Figure 2.3: Phone bill Analyzer App .....	19
Figure 3.1: use case diagram.....	25
Figure 3.2: Class diagram .....	26
Figure 3.3: sequence diagram.....	خطأ! الإشارة المرجعية غير معروفة.
Figure 3.4: Activity Diagram .....	31
Figure 3.5: Context flow diagram.....	32
Figure 3.6: Dataflow diagram.....	33
Figure 3.7: ER diagram .....	34
Figure 3.8: User table.....	35
Figure 3.9: user_details table.....	35
Figure 3.10: user_budget table .....	35
Figure 3.11: user_targets table .....	36
Figure 3.12: electricity_bill table.....	36
Figure 3.13: phone_bill table .....	36
Figure 3.14: gas_bill table .....	37
Figure 3.15: water_bill table .....	37
Figure 3.16: Welcome page .....	38
Figure 3.17: Home page .....	39
Figure 3.18: My budget page .....	40
Figure 3.19: My bills page .....	41
Figure 3.20: Add New Bills page.....	42
Figure 3.21: Report page.....	43
Figure 4.1: Landing page hero section .....	65
Figure 4.2: Landing page features section .....	65
Figure 4.3: Landing page review section .....	66
Figure 4.4: Landing page our team section .....	66
Figure 4.5: Landing page contact section.....	67
Figure 4.6: Landing page map and footer .....	67
Figure 4.7: Register interface .....	73
Figure 4.8: Login interface .....	75
Figure 4.9: Add bill interface .....	77
Figure 4.10: View bill interface .....	78
Figure 4.11: Danger notification interface .....	81
Figure 4.12: Scuccess notification interface .....	81
Figure 4.13: Main dashboard interface .....	82
Figure 4.14: Budget interface.....	84
Figure 4.15: Reports interface .....	85
Figure 4.16: Notification interface .....	86

## **LIST OF ABBREVIATIONS**

bBills.....(Budget Bills)

CGI.....(Common Gate Interface)

MVC.....(Model-Controller-View)

APK.....(Android Package Kit)

## **1. CHAPTER 1: INTRODUCCION**

### **1.1. INTRODUCTION**

In many cases, citizens may not be always aware of the monthly consumption of resources, as they face problems with high bills without realizing it, and they may realize it at the time of paying the bill!

In order to take advantage of the technological development taking place in data analysis and extracting the results that are of interest from it, we are planning to create a website with a data analysis system to work on analyzing the user's bills and giving them alerts when they increase or decrease significantly. Such system will help them in managing their bills expenses and consumption efficiently.

### **1.2. PROBLEM STATEMENT**

The head of the family always needs to calculate their monthly budget, in order to know its appropriate distribution, and in some cases, they do not realize the amount of their monthly consumption, as the surprise comes at the time of paying the bills.

Therefore, they strive to make consumption within a specific field, in order to avoid any significant increase outside the normal limit, so that consumption is in accordance with the budget.

### **1.3. PROBLEM SOLUTION**

To solve the above problem, we proposed a system that analyzes bills such as (electricity, gas, water, phone , etc) for citizens in the Kingdom of Saudi Arabia. Through this system, users' data is analyzed and she alerted when the value of her bills increases or decreases significantly, allowing users to better manage their financial resources.

#### **1.4. AIMS OF THE PROJECT**

This project aims to analyze household bills for users and give them alerts in the event that a bill is received at a high or low rate, as it helps them manage their financial affairs better by using the analysis and study of their bills data.

#### **1.5. OBJECTIVES OF THE PROJECT**

1. Building a system that analyzes bills.
2. Notifying users when bill value increases or decreases significantly.
3. Allowing users to set monthly budget to monitor expenses.

#### **1.6. SCOPE OF THE PROJECT**

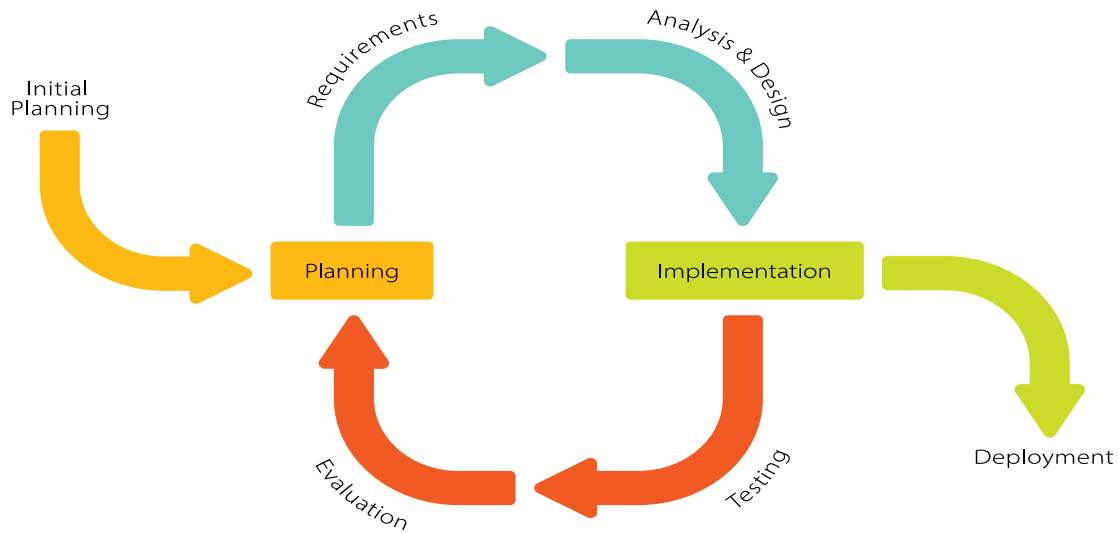
The system is proposed to be used in Saudi Arabia to help citizens manage their budget better, and be aware of their monthly usage of resources, the bills are (electricity, gas, water, phone, etc).

#### **1.7. SIGNIFICANCE OF THE PROJECT (MOTIVATION)**

The importance of the project lies in the financial assistance of citizens in the monthly financial management, and to stay aware of their consumption of resources to remain in within their budget also provide target tracking for bills .

#### **1.8. SOFTWARE DEVELOPMENT METHODOLOGIES**

The system will be built according to the Agile methodology, where it helps to develop the system through iterative steps, where available program analysis facilities. It includes analysis of structure, modularity, usability, reliability, efficiency, and goal achievement. Here are the implementation steps:



**Figure 1.1: Agile methodology**

### **1.8.1. Planning**

Scope of the project, project organization, plans, estimated cost, risks management and time budget and risks are introduced in this section.

### **1.8.2. Analysis**

It describes the requirements that are needed to develop the system. It includes the specification requirements (functional and non-functional). Also, use cases that show the interactions between the user and the system and the requirements analysis. In addition, the databases design of the system. Moreover, the requirements needed for the project are going to be described and organized. Also, analysis of the system, and a description of the process model.

### **1.8.3. Design**

This phase is for describing the design of the system. It involves internal and external user interfaces, a brief overview of the code and relationships between the objects. In addition, a full design presentation of the project with every necessary detail related to the system.

### **1.8.4. Implementation**

This phase shows the real implementation of the system in terms of making the system ready to be used by the users.

#### **1.8.5. Testing**

This phase shows the testing of the whole project after implementation with all the tools, and methodologies used.

#### **1.8.6. Evaluation**

This phase is to ensure that the system is working smoothly without any problem. Also, support for the user is presented in this phase.

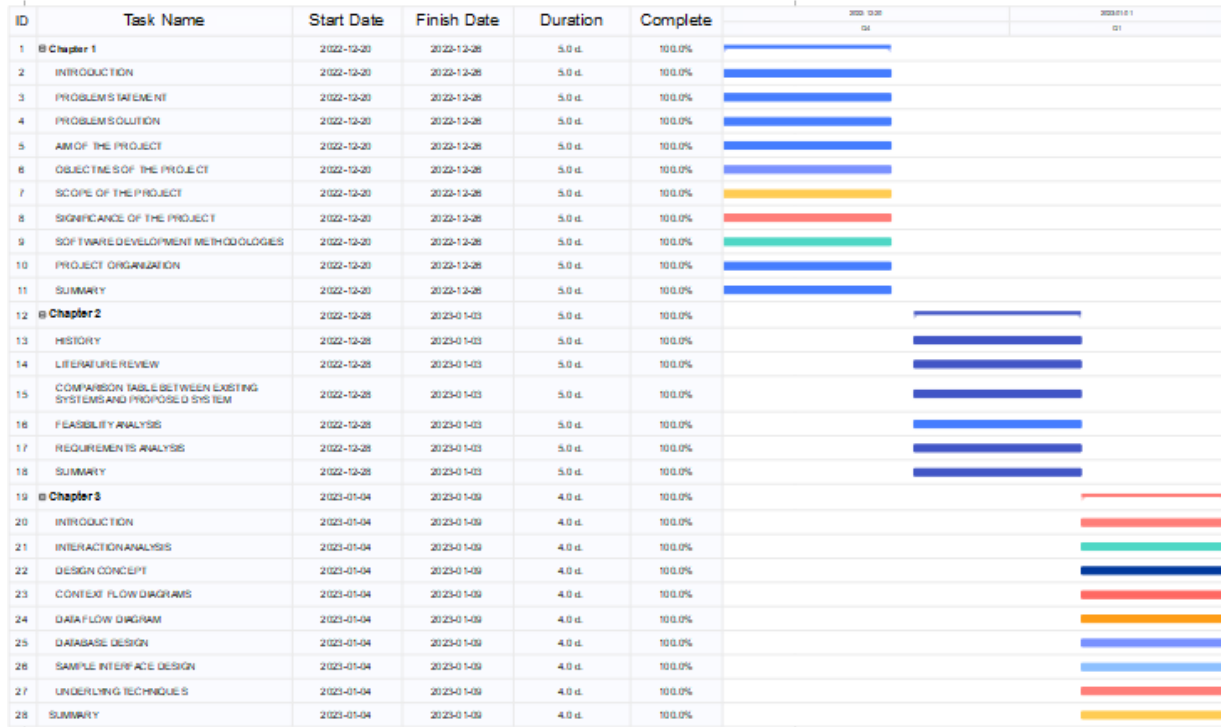
### **1.9. PROJECT ORGANIZATION**

#### **1.9.1. Member's Tasks**

**Table 1.1: Member's Task**

<b>Task</b>	<b>Student's name</b>
<b>Chapter 1</b>	Farah Hamad Mohammed Aldossary Reuof Ali Ali Alqahtani Lubna Abdullah Ali Alqahtani
<b>Chapter 2</b>	Lamees Abdullah Saad Al Shahrani Amani Abdullah Ayed Alqahtani Nada Saad Al Kuraydis Shahad Abdullah Ali Al_Sreei
<b>Chapter 3</b>	<b>All Students</b>

#### **1.9.2. Gantt Chart**



**Figure 1.2: Gantt Chart**

## 1.10. SUMMARY

In this chapter, we studied the broad outlines of the problem, and presented the proposed solution that we will work on, which is to build a website that alerts the user in the event of an increase or decrease in resource consumption, to help them organize and manage their financial affairs.

## **2. CHAPTER 2: BACKGROUND AND LITERATURE REVIEW**

### **2.1. LITERATURE REVIEW**

#### **1. AlKahraba app**

It is an application launched by the Electricity Company in the Kingdom of Saudi Arabia, where it is easy for citizens to pay electricity bills and see all the details of their accounts from previous bills, their consumption over different periods of time. Clients can subscribe to or cancel the service through it [1].

Advantages:

- consumption graph
- User will have a APK file in his memory.
- View all details
- Secure
- You can set a known, fixed monthly budget of your power needs
- Notification of payment of bill

Disadvantages:

- Specific to electricity bills only
- The negatives, the clearance numbers after 12 months may be shocking to the customer and he becomes demanding a high amount that is difficult to pay in one month.
- The inability to upload some electronic requests through the application, it is only done through the website of the Saudi Electricity Company.



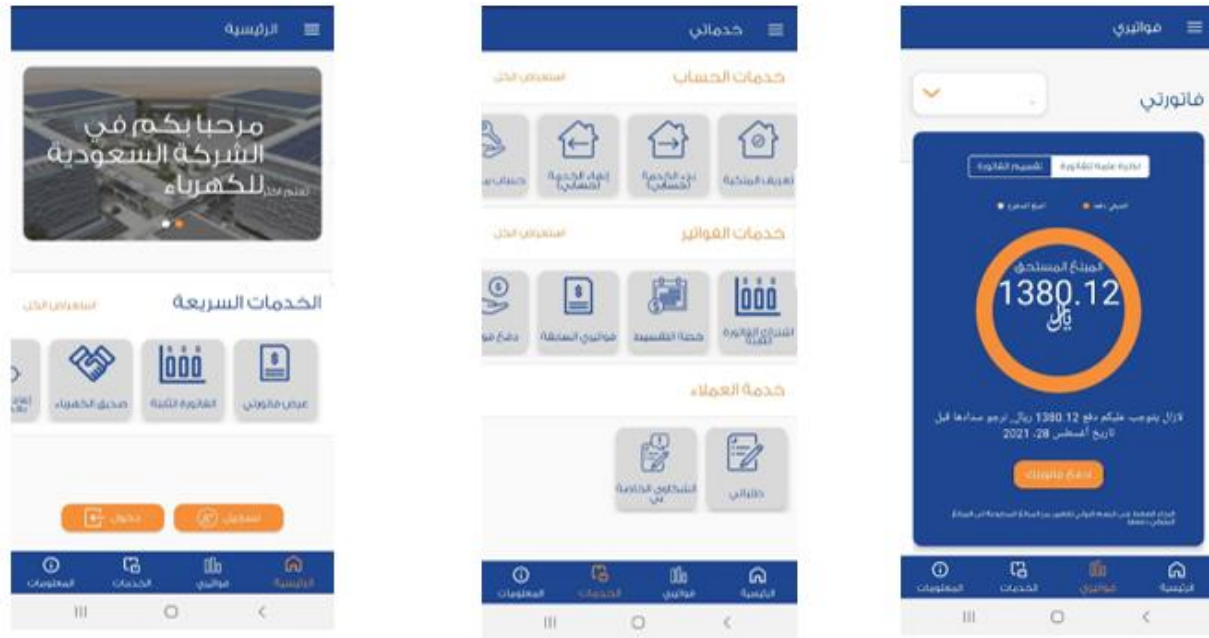


Figure 2.1: Alkahraba app

## 2. MINT App

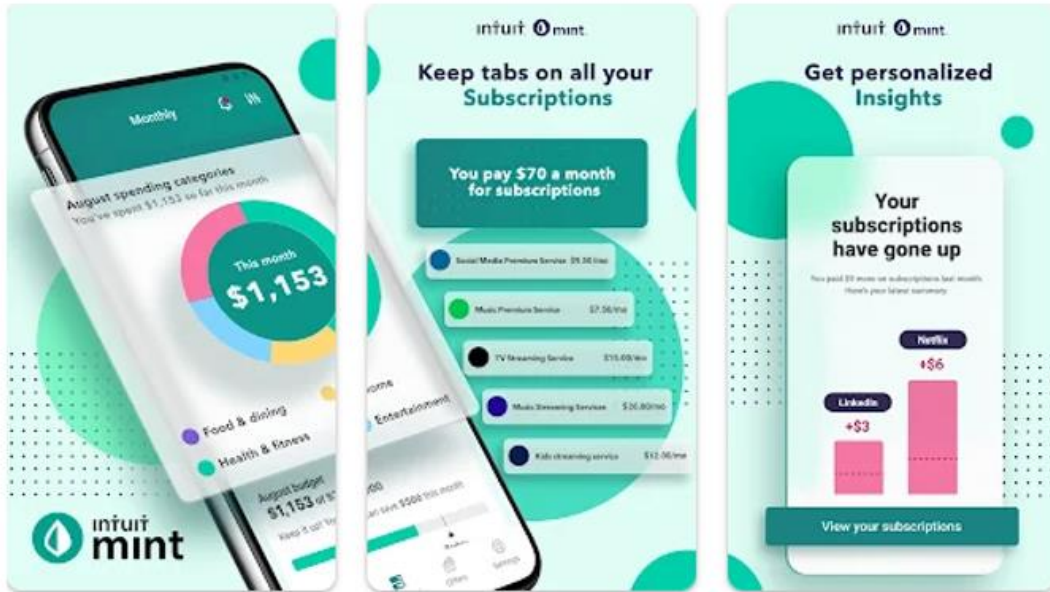
Mint is a budgeting app that offers a lot of features at no cost, including income and expense planning and tracking, bill negotiation and even credit monitoring. [2].

Advantages:

- Secure
- Syncs to a diverse set of financial accounts
- Includes alert and reminder tools
- Free credit monitoring services
- Calculates net worth

Disadvantages:

- Takes time to set up
- Occasional account connection issues
- You can't get all of the app's features without paying a monthly fee



**Figure 2.2: MINT app**

### **3. Phone bill Analyzer App**

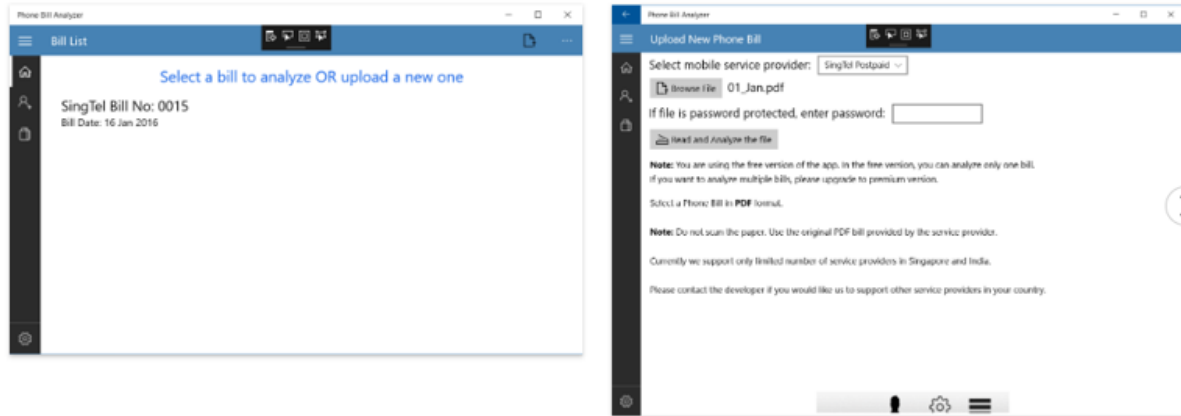
An application that helps analyze the phone bill, where the user can identify the exact amount of the call for each contact, and a group of contacts as specified in the user's contact list. With this app user can upload bill and assign contacts. The app supports postpaid mobile bills only [3].

#### **Advantages**

- Detailed call quality metrics
- Expanded range of local calling plans
- It is a phone memo that helps to save contacts.
- Analyzing and paying the telecom bill.

#### **Disadvantages**

- Calling plan is still limited.
- Analyze and control one type of billing.



**Figure 2.3: Phone bill Analyzer App**

## 2.2. COMPARISON TABLE BETWEEN EXISTING SYSTEMS AND PROPOSED SYSTEM

**Table 2.1: Comparison Table with existing systems**

Services	Alkahraba app	Mint app	Phone bill analyser app	Our System
Analysis bills	Yes	No	Yes	Yes
View previous bills (dashboard)	Yes	Yes	No	Yes
The type of bills being analyzed	Electricity bills	Daily bills	Phone bills	Electricity, Water, Gas, Phone
Users set monthly budget	No	Yes	No	Yes
Notify user about increase or decrease the bill	No	Yes	Yes	Yes
Target tracking	No	No	No	Yes
Predictions	Yes	No	No	Yes

## **2.3. FEASIBILITY ANALYSIS**

It is necessary to determine if our project is feasible or not. Project feasibility is assessed in two principal ways: technically, and economically. A project must be feasible in all these ways to be developed, and resources must be discussed in relationship to these areas.

### **2.3.1. TECHNICAL FEASIBILITY**

Below we list some technical needs that will be the reason for implementing the project. The development of this project requires two types of requirements, the physical requirements, which express computer hardware and software requirements. And software requirements express the necessary software in order to create applications in their proper form.

#### **2.3.1.1. SOFTWARE REQUIREMENT**

We will use these programs:

- For website:
  - ❖ HTML, CSS, PHP (Visual studio code)
  - ❖ MySQL (Xampp server)
- For analysis system:
  - ❖ Python (Anaconda)

#### **2.3.1.2. HARDWARE REQUIREMENT**

As for the physical requirements:

- Developer's laptop with at least 8th generation Core i5 processor.
- 8 GB of random-access memory (RAM).
- Windows 10, plus an internet connection.
- A graphics card with a capacity of at least 2 gigabytes.
- Storage space of at least 100 GB.

### **2.3.2. ECONOMICAL FEASIBILITY**

The following shows the financial cost of the project:

#### **2.3.2.1. SOFTWARE REQUIREMENT COST**

All the previously mentioned software is open-source software, and all software is free. can be downloaded with all the libraries we need. So, there will be no cost.

#### **2.3.2.2. HARDWARE REQUIREMENT COST**

The price of a computer with the previously specified specifications and hardware requirements ranges from 2795 SAR to 3000 SAR.

### **2.4. REQUIREMENTS ANALYSIS**

A requirement is a description of the service that the program should provide. Requirements describe the solution to be developed, including its functionality, interfaces, design, and user experience. They are usually formulated by the client or stakeholders, which are the people who are involved in or affected by the project and therefore interested in its success.

For optimal implementation requirements are divided into functional requirements and non-functional requirements. The following are provided for these requirements as applicable.

#### **2.4.1. FUNCTIONAL REQUIREMENT**

It is the data of what services the system should provide, how the system should react to certain inputs and how the system should behave in certain situations. It describes the functions or services of the system, and is dependent on the type of software, the expected users, and the type of system where the software is used. The following are the functional requirements for an bills analysis website:

- **bBill Requirements**

- Data analysis: The system is responsible for analyzing the data using the Python programming language that has a wide range of libraries for numerical computation and data manipulation.
- Send notification: The system sent notifications if the result of the analysis gives too much or too little consumption.
- Target tracking: The system shows with progress bars how much you've budgeted and saving on each bills type.
- Prediction: Built linear regression model, it is one of the most-used regression algorithms in Machine Learning. A significant variable from the data set is chosen to predict the output variables (future values)

- **User Requirements**

- Registration: The system allows the user to register in the system by filling out his personal data and billing information form.
- Login: The system allows the user to log into the system by entering his username and password.
- Logout: The system allows the logged in user to log out of the site.
- Set / Update monthly budget: The site allows the user to set or update the monthly budget.
- Add / remove bills: the website allows the user to add or remove his bills.
- Receive notification: The site sends the user a notification in the event of a significant increase or decrease in consumption.
- Receive analysis bills result: The system analyzes data (bills) and displays it as a data visualization (Dashboard).
- Add saving target: The system allows the user to tracking spending and savings target by add saving target.

#### **2.4.2. NON-FUNCTIONAL REQUIREMENT**

This system contains some non-functional factors that evaluate the activity and performance of the provided services, which are described below:

- Ease of use: The design should be elegant, clear and uncomplicated.
- Reliability: The information in the site must be reliable and accurate.
- Availability: The website must be available at any time and serve the largest number of users at the same time.
- Maintainability: to allow the development of website, the addition of new features, and the development of services.
- Security: The site must be protected from unauthorized access to the system and its stored data. This system will be secure because the system will use password hashing and it will be difficult to detect.

## 2.5. SUMMARY

In this chapter, applications that are similar to our website were studied to highlighted their features and limitation and then they were compared with our proposed system.

Next, we worked on studying the requirements of the project and the devices, and determining the permissions of each user in the system, with the economic feasibility and the total cost of the hardware and software that the project might need to be developed.

### **3. CHAPTER 3 SYSTEM ANALYSIS AND SYSTEM DESIGN**

#### **3.1. INTRODUCTION**

In this chapter, we analyze and understand the system completely through UML diagrams, where the database structure will be determined in which the data necessary to operate the system will be stored.

#### **3.2. INTERACTION ANALYSIS**

After starting to analyze the system, we can understand and clarify the basic functions of the system. Where work is then done to explain the system through UML (Unified Modeling Language) diagrams, through which illustrations are made that help to understand the operation of the system.

##### **3.2.1. UML DIAGRAM**

###### **3.2.1.1. USE-CASE DIAGRAM**

To fully understand the functioning of the system, you must define and understand the tasks it performs to process user requests in order to display the results.

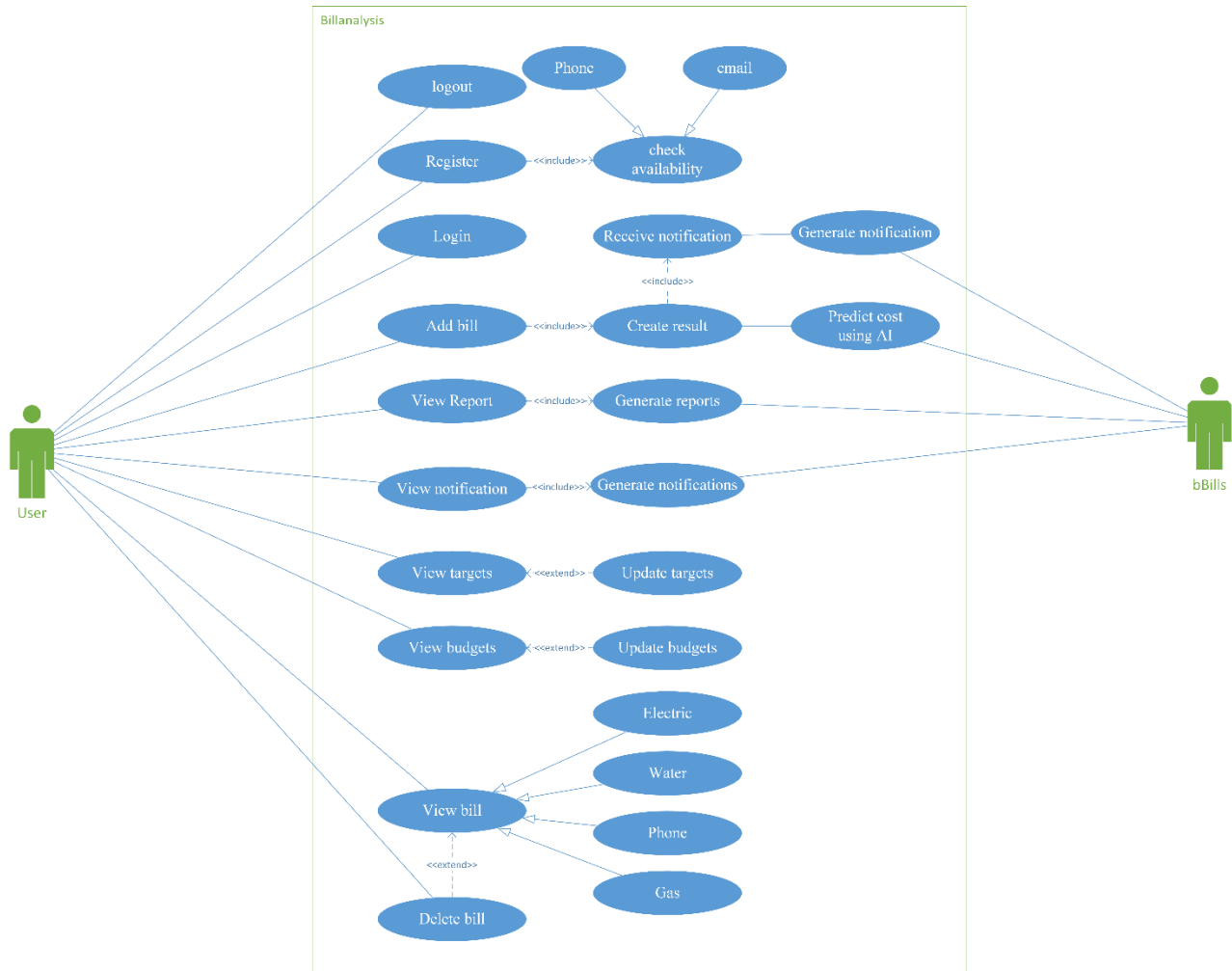
Within UML diagrams, many diagrams are available to describe the behavior within a system, and to understand its functions and details. Use case diagram is an important diagram for understanding the functionality of the components of the system.

Among the goals of the use case diagram are:

- Getting an external view of the system.
- Clarifying the interaction between the requirements of the components of the system.
- Collecting system requirements.



Figure 3.1 shows the proposed system use case diagram:



**Figure 3.1: use case diagram**

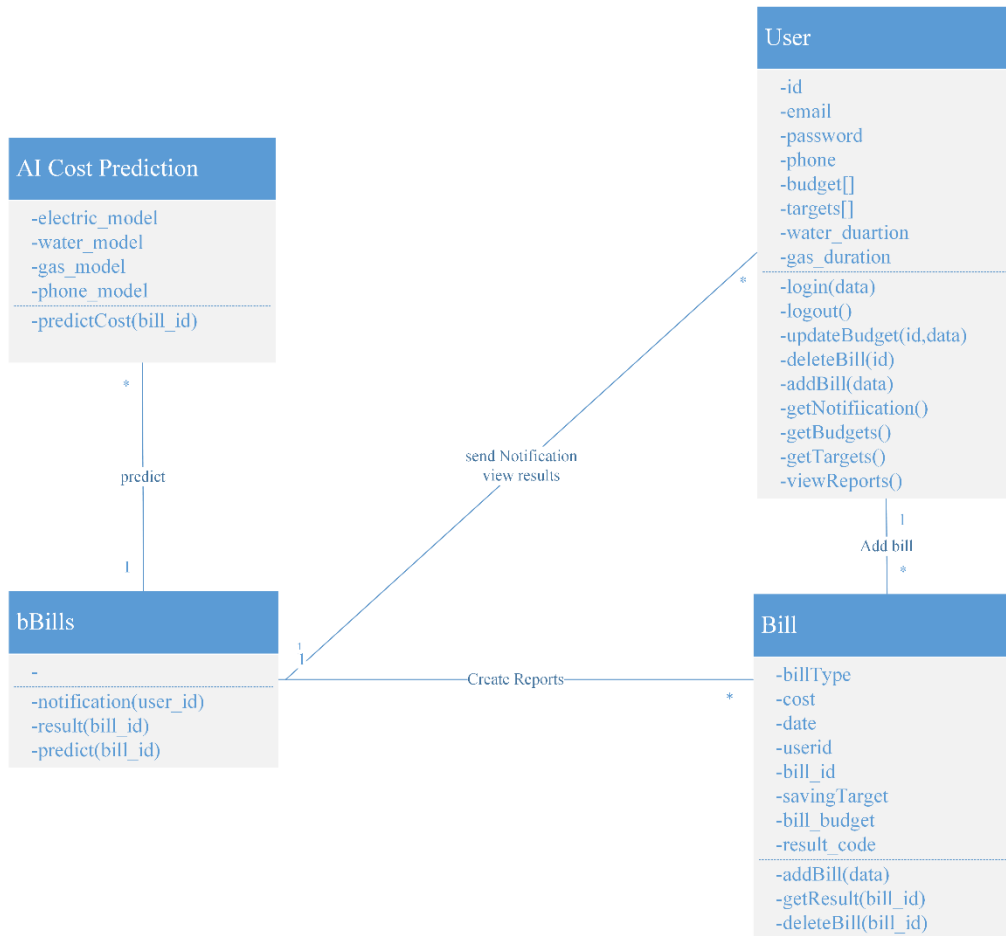
### 3.2.1.2. CLASS DIAGRAM

A UML class diagram is a graphical diagram that is used to construct a visualization describing the structure of a system by showing: the classes, their attributes, the operations performed by each part of the system, and the relationships between things.

Objectives of class diagram are summarized below:

- Modeling the objects that make up the system.
- Show relationships between objects.
- Describe what these objects and their services do.

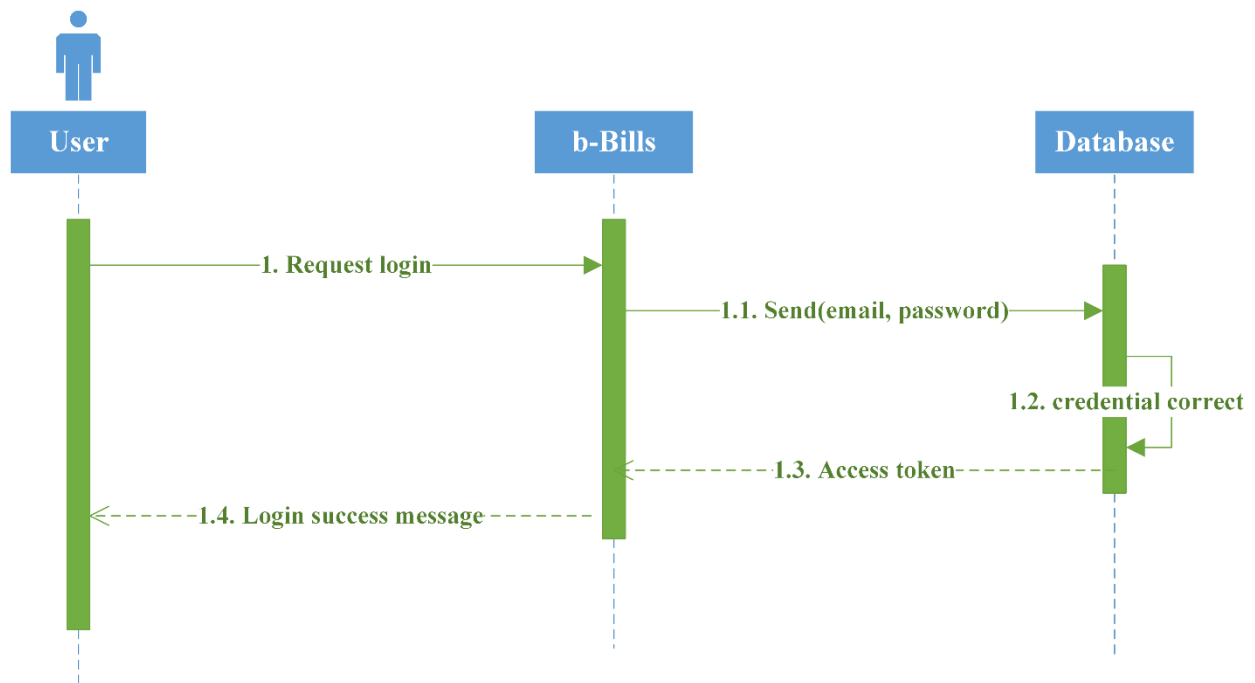
Figure 3.2 shows the class diagram related to our proposed system.



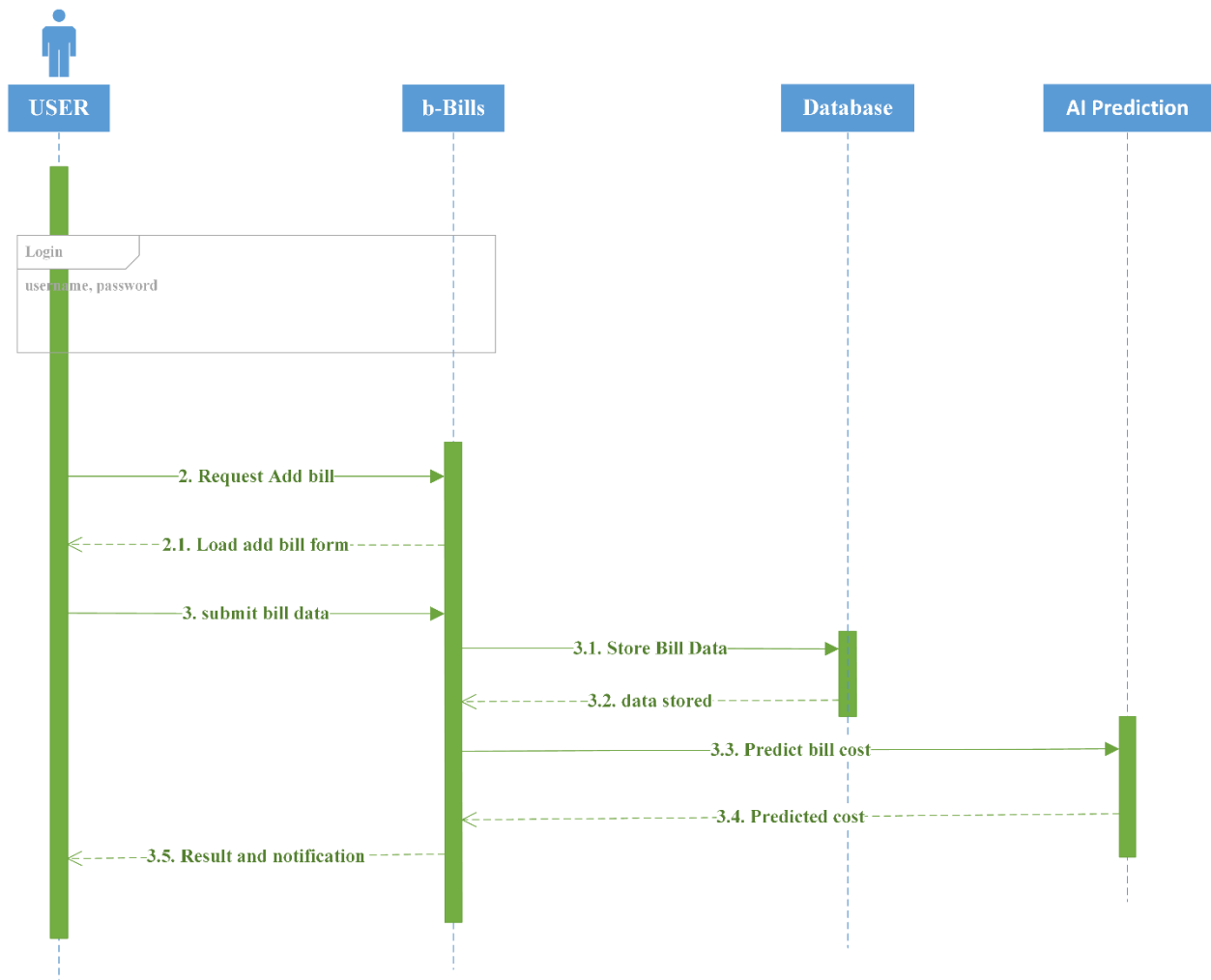
**Figure 3.2: Class diagram**

### 3.2.1.3. SEQUENCE DIAGRAM

A sequence diagram is a type of UML diagram that shows how objects in a system or class interact with each other, and the interactions are shown sequentially for each event that the user can perform, in order to understand the requirements of the system and how it works.



**Figure 3.3: Login sequence diagram**



**Figure 3.4: Add bill sequence diagram**

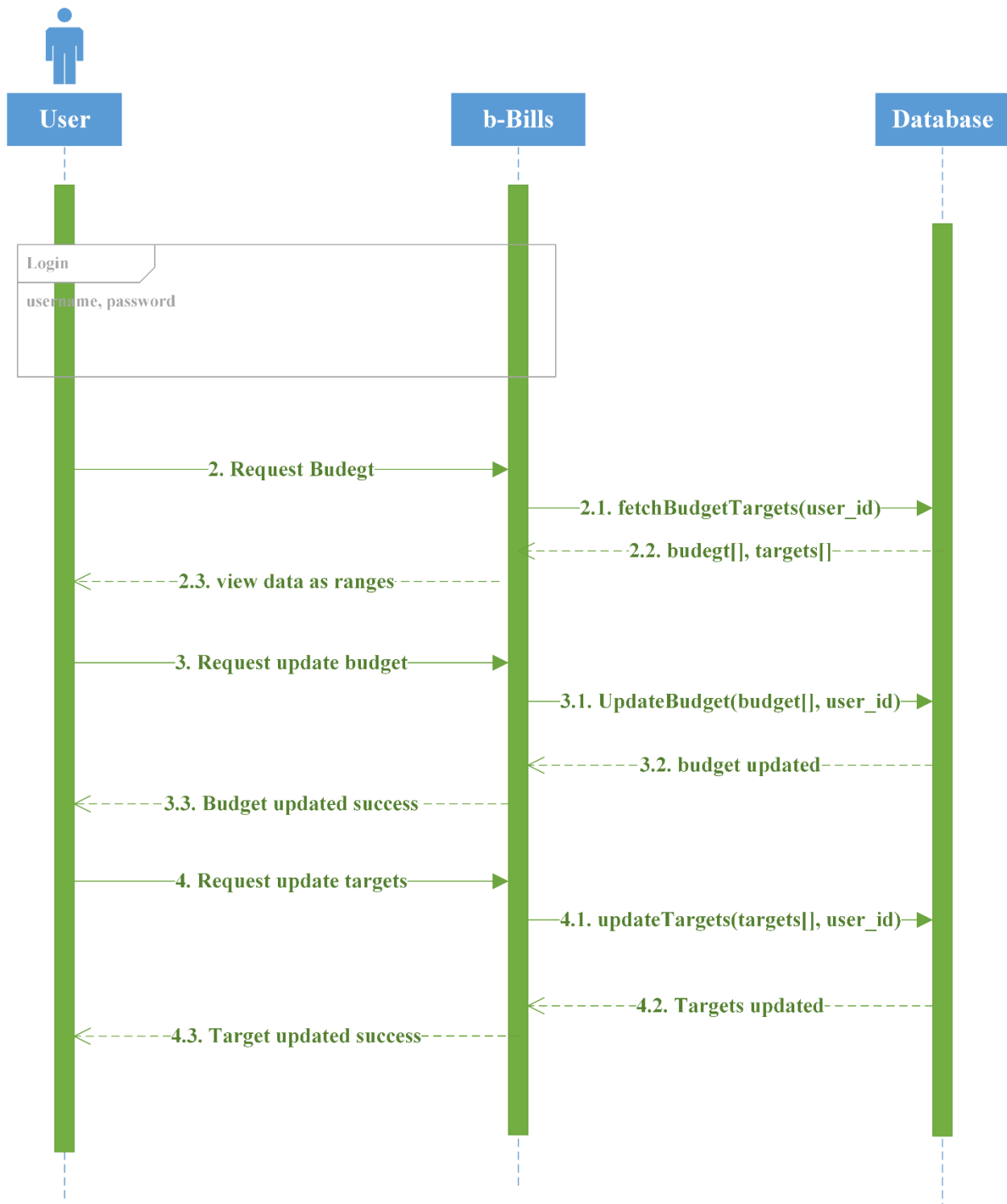
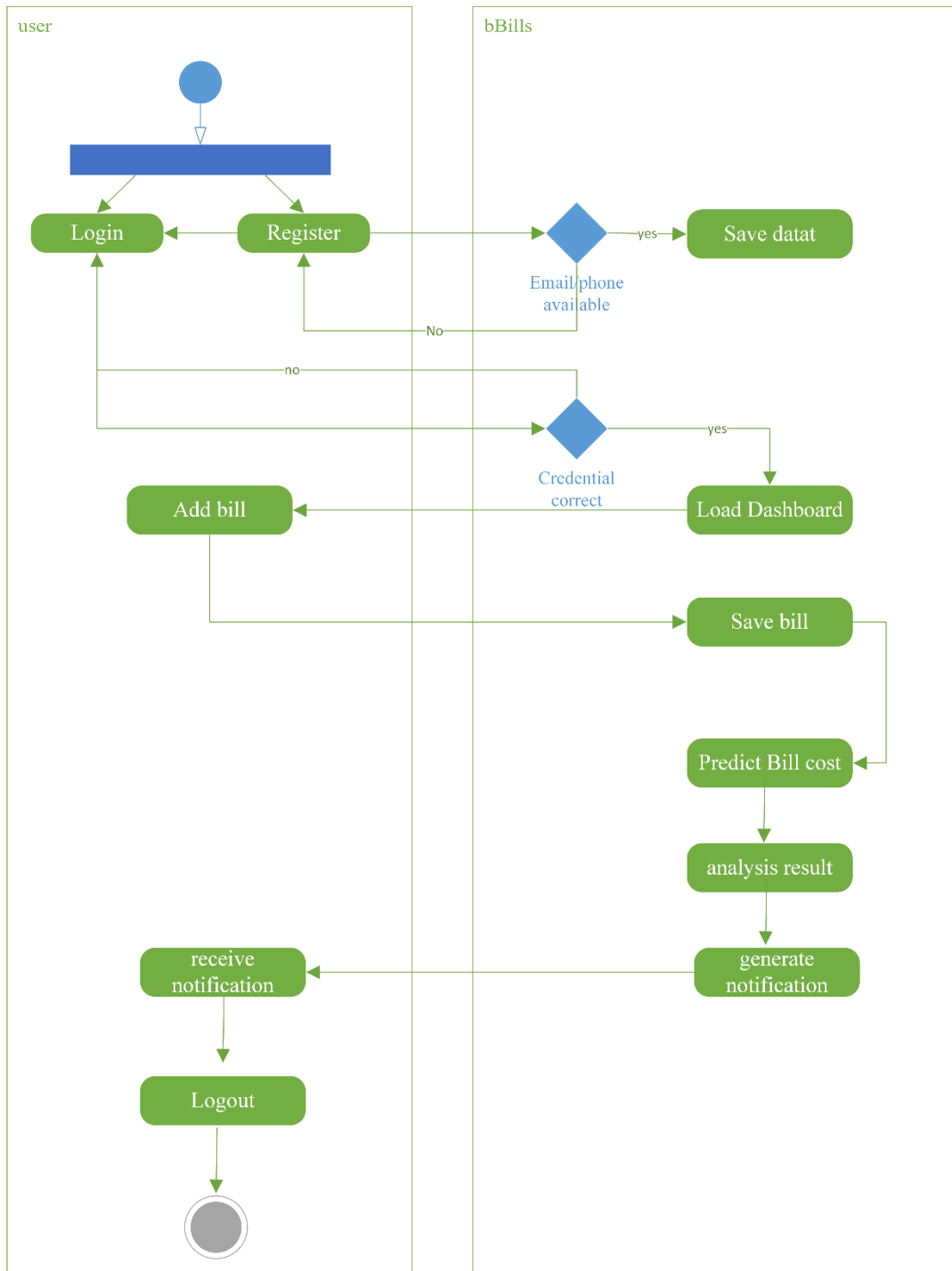


Figure 3.5: Update budget and targets sequence diagram

#### **3.2.1.4. ACTIVITY DIAGRAM**

The activity diagram is used to describe the automatic aspects of the system. It shows the flow of activity and the transition of the system from one state to another, where the activities are described as events within the system.

Figure 3.6 shows system activity diagram.



**Figure 3.6: Activity Diagram**

### 3.3. DESIGN CONCEPT

The entire system requirements will be studied for its design, and to determine the functional and non-functional requirements necessary to operate the system, as the system is

represented by UML diagrams that give a full explanation of the system and its performance, with its complete components, as the system codes will be built according to these diagrams.

### 3.4. CONTEXT FLOW DIAGRAMS

Context diagrams show the interaction between the system and other actors (external agents) with which the system is designed to interact. System context diagrams can be useful in understanding the context that a system will be a part of.

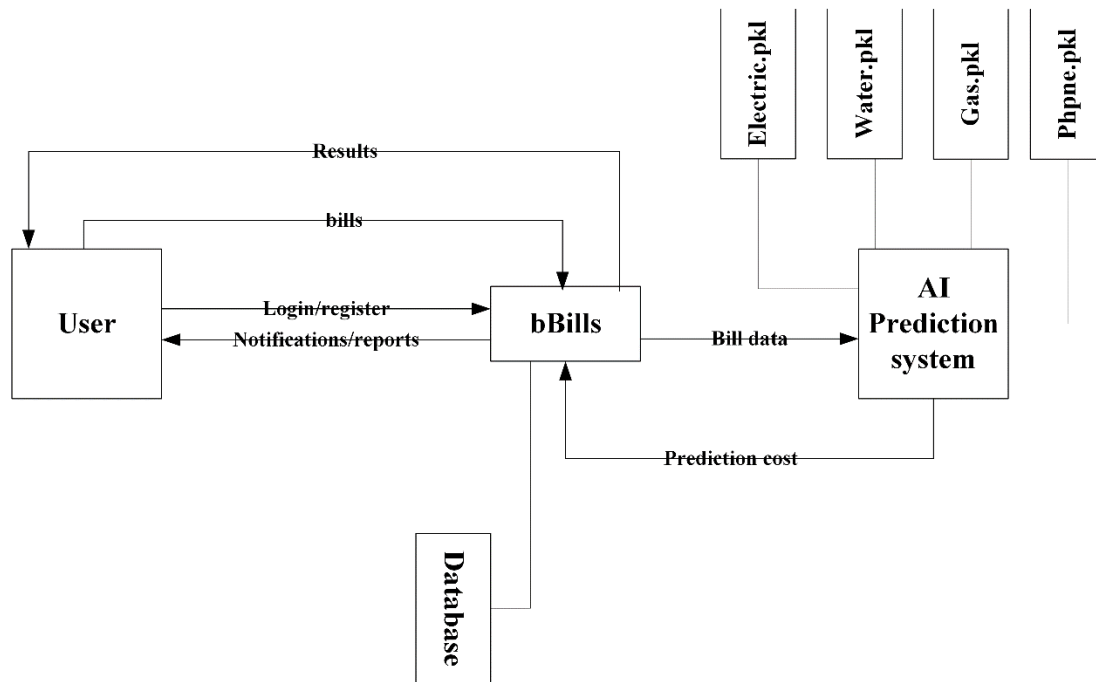
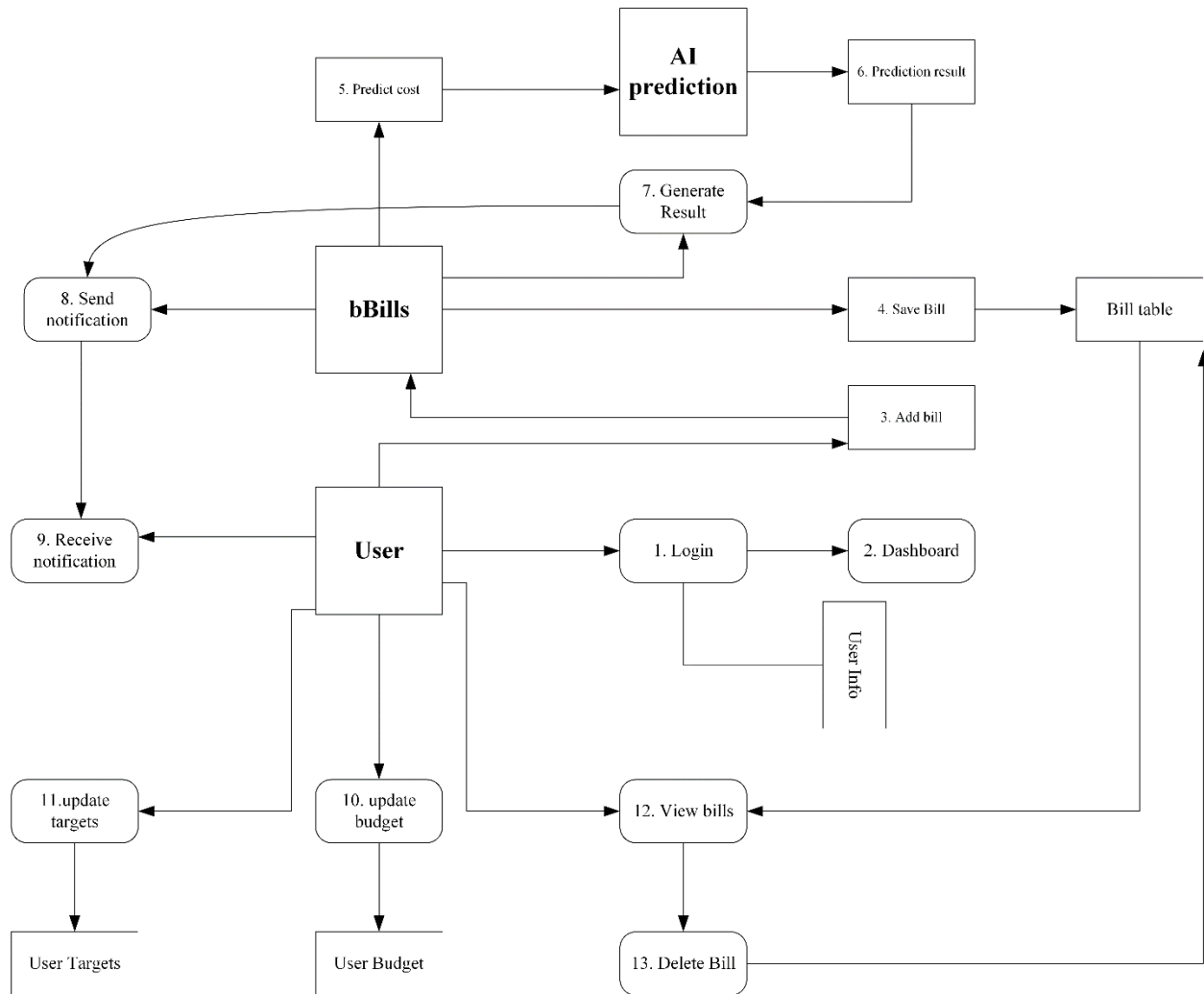


Figure 3.7: Context flow diagram

### 3.5. DATA FLOW DIAGRAM

A data flow diagram is a cm that expresses the flow of data between different processes within a system. Where the flow of information used to transfer data from input to output is represented.



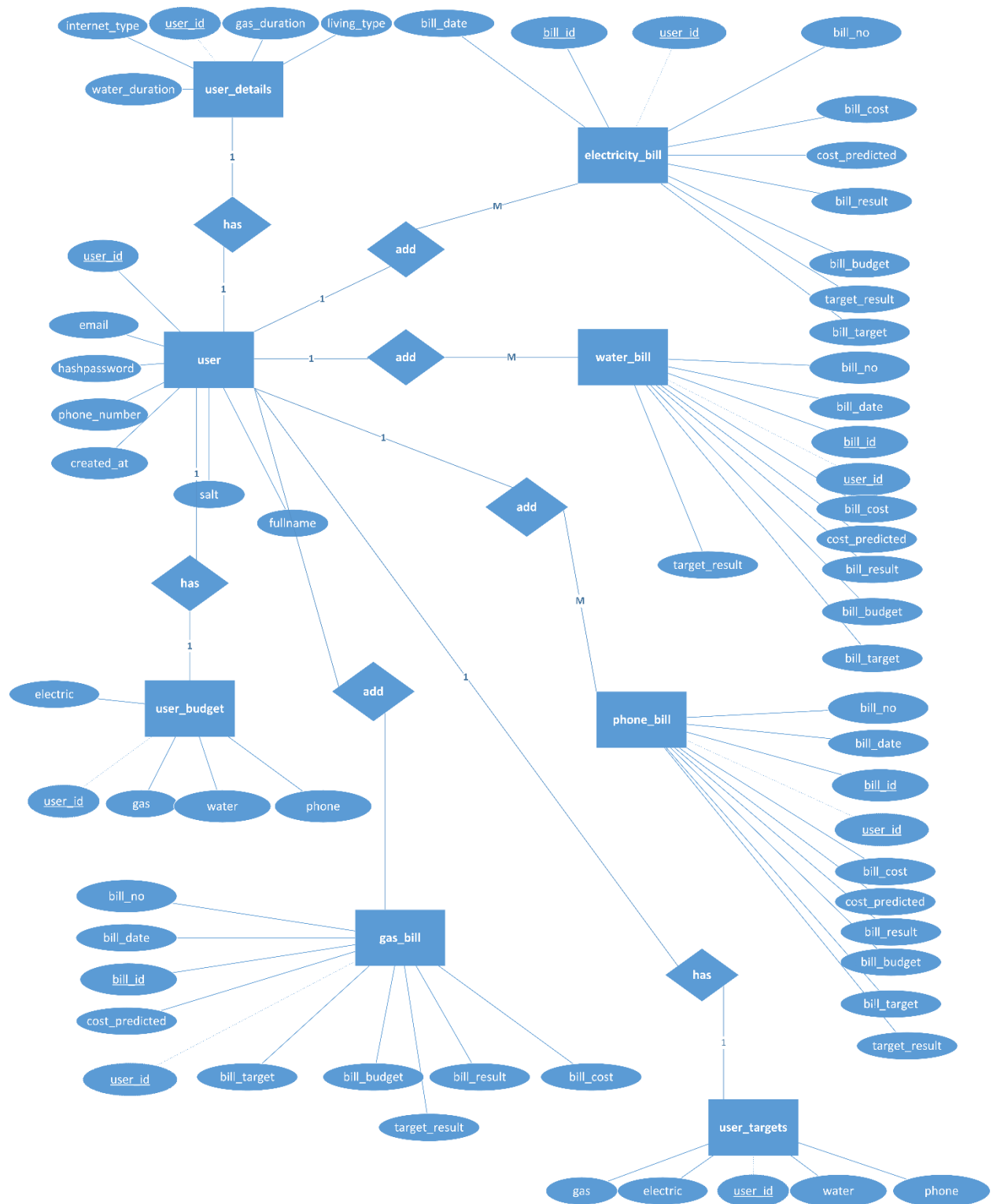


**Figure 3.8: Dataflow diagram**

## 3.6. DATABASE DESIGN

### 3.6.1. ER DIAGRAM

It is an analytical diagram that shows the relationship of system components with each other within the system. This diagram is often used to clarify the object relations between the parts and components of the system.



**Figure 3.9: ER diagram**

### 3.6.2. DATABASE TABLES


	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
<input type="checkbox"/>	1	user_id 	int(11)			No	None		AUTO_INCREMENT
<input type="checkbox"/>	2	fullname	varchar(200)	utf8_general_ci		No	None		
<input type="checkbox"/>	3	email	varchar(200)	utf8_general_ci		No	None		
<input type="checkbox"/>	4	hashpassword	varchar(200)	utf8_general_ci		No	None		
<input type="checkbox"/>	5	salt	varchar(10)	utf8_general_ci		No	None		
<input type="checkbox"/>	6	phone_number	int(10)			No	None		
<input type="checkbox"/>	7	created_at	datetime			No	None		

Figure 3.10: User table


	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
<input type="checkbox"/>	1	id 	int(11)			No	None		AUTO_INCREMENT
<input type="checkbox"/>	2	user_id	int(11)			No	None		
<input type="checkbox"/>	3	living_type	tinyint(4)			No	None		
<input type="checkbox"/>	4	internet_type	tinyint(4)			No	None		
<input type="checkbox"/>	5	gas_duration	tinyint(4)			No	None		
<input type="checkbox"/>	6	water_duration	tinyint(4)			No	None		

Figure 3.11: user\_details table


	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
<input type="checkbox"/>	1	id 	int(11)			No	None		AUTO_INCREMENT
<input type="checkbox"/>	2	user_id	int(11)			No	None		
<input type="checkbox"/>	3	electricity	float			No	None		
<input type="checkbox"/>	4	phone	float			No	None		
<input type="checkbox"/>	5	water	float			No	None		
<input type="checkbox"/>	6	gas	float			No	None		

Figure 3.12: user\_budget table

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
<input type="checkbox"/>	1	id 🔑	int(11)			No	None		AUTO_INCREMENT
<input type="checkbox"/>	2	user_id	int(11)			No	None		
<input type="checkbox"/>	3	electricity	float			No	None		
<input type="checkbox"/>	4	phone	float			No	None		
<input type="checkbox"/>	5	water	float			No	None		
<input type="checkbox"/>	6	gas	float			No	None		


**Figure 3.13: user\_targets table**

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
<input type="checkbox"/>	1	electricity_bill_id 🔑	int(11)			No	None		AUTO_INCREMENT
<input type="checkbox"/>	2	user_id	int(11)			No	None		
<input type="checkbox"/>	3	bill_no	varchar(200)	utf8_general_ci		No	None		
<input type="checkbox"/>	4	bill_date	date			No	None		
<input type="checkbox"/>	5	bill_cost	float			No	None		
<input type="checkbox"/>	6	predicted_cost	float			No	None		
<input type="checkbox"/>	7	electricity_bill_result	tinyint(4)			No	None		
<input type="checkbox"/>	8	electricity_budget	float			No	None		
<input type="checkbox"/>	9	electricity_target	float			No	None		
<input type="checkbox"/>	10	target_result	tinyint(4)			No	None		
<input type="checkbox"/>	11	created_at	datetime			No	None		


**Figure 3.14: electricity\_bill table**

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
<input type="checkbox"/>	1	phone_bill_id 🔑	int(11)			No	None		AUTO_INCREMENT
<input type="checkbox"/>	2	user_id	int(11)			No	None		
<input type="checkbox"/>	3	bill_no	varchar(200)	utf8_general_ci		No	None		
<input type="checkbox"/>	4	bill_date	date			No	None		
<input type="checkbox"/>	5	bill_cost	float			No	None		
<input type="checkbox"/>	6	predicted_cost	float			No	None		
<input type="checkbox"/>	7	phone_bill_result	tinyint(4)			No	None		
<input type="checkbox"/>	8	phone_budget	float			No	None		
<input type="checkbox"/>	9	phone_target	float			No	None		
<input type="checkbox"/>	10	target_result	tinyint(4)			No	None		
<input type="checkbox"/>	11	created_at	datetime			No	None		

**Figure 3.15: phone\_bill table**

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
<input type="checkbox"/>	1	gas_bill_id 	int(11)			No	None		AUTO_INCREMENT
<input type="checkbox"/>	2	user_id	int(11)			No	None		
<input type="checkbox"/>	3	bill_no	varchar(200)	utf8_general_ci		No	None		
<input type="checkbox"/>	4	bill_date	date			No	None		
<input type="checkbox"/>	5	bill_cost	float			No	None		
<input type="checkbox"/>	6	predicted_cost	float			No	None		
<input type="checkbox"/>	7	gas_bill_result	tinyint(4)			No	None		
<input type="checkbox"/>	8	gas_budget	float			No	None		
<input type="checkbox"/>	9	gas_target	float			No	None		
<input type="checkbox"/>	10	target_result	tinyint(4)			No	None		
<input type="checkbox"/>	11	created_at	datetime			No	None		

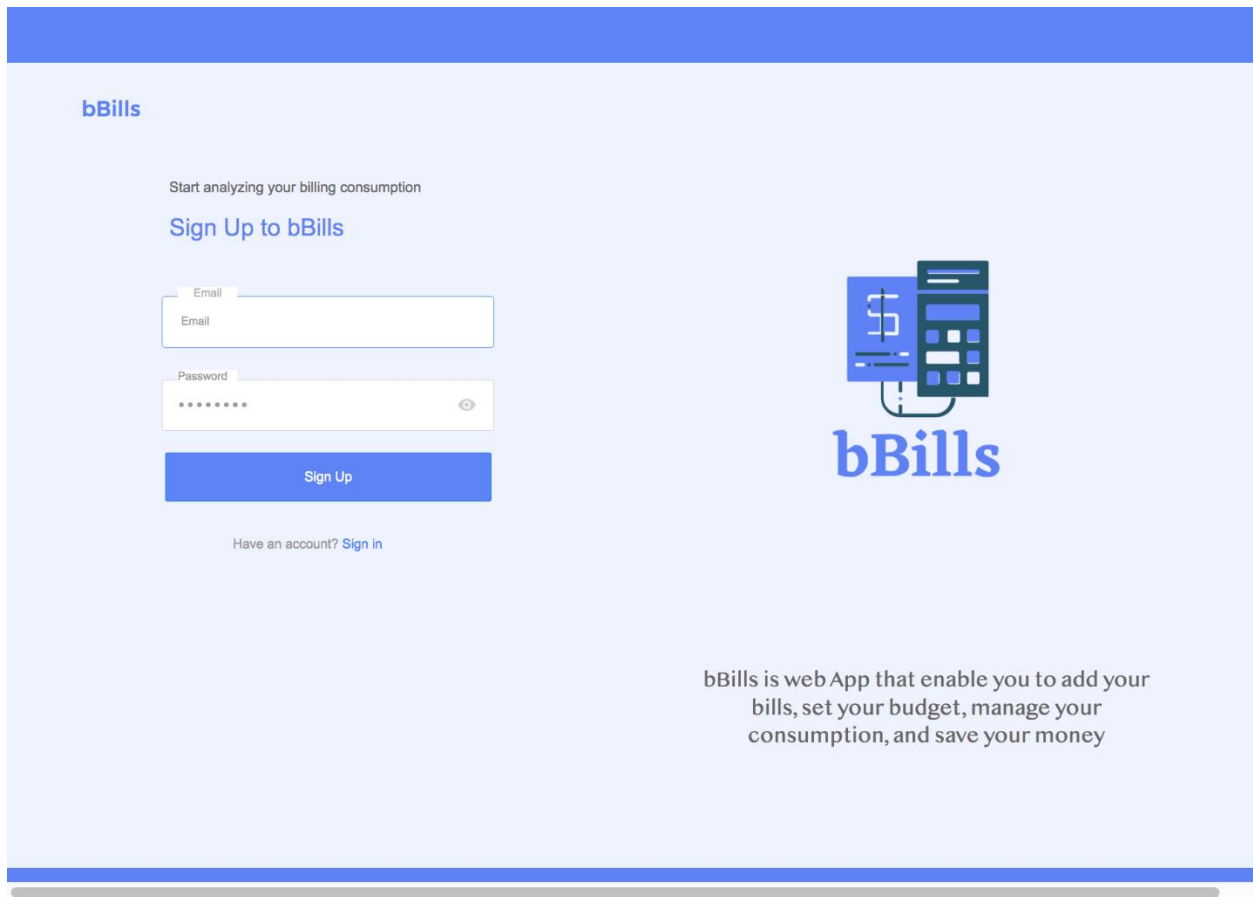
**Figure 3.16: gas\_bill table**

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
<input type="checkbox"/>	1	water_bill_id 	int(11)			No	None		AUTO_INCREMENT
<input type="checkbox"/>	2	user_id	int(11)			No	None		
<input type="checkbox"/>	3	bill_no	varchar(200)	utf8_general_ci		No	None		
<input type="checkbox"/>	4	bill_date	date			No	None		
<input type="checkbox"/>	5	bill_cost	float			No	None		
<input type="checkbox"/>	6	predicted_cost	float			No	None		
<input type="checkbox"/>	7	water_bill_result	tinyint(4)			No	None		
<input type="checkbox"/>	8	water_budget	float			No	None		
<input type="checkbox"/>	9	water_target	float			No	None		
<input type="checkbox"/>	10	target_result	tinyint(4)			No	None		
<input type="checkbox"/>	11	created_at	datetime			No	None		

**Figure 3.17: water\_bill table**

## 3.7. SAMPLE INTERFACE DESIGN

### 1. WELCOME PAGE



The image shows a web application interface for 'bBills'. It features a light blue background with a white sign-up form on the left and a large illustration on the right. The form includes fields for 'Email' and 'Password', a 'Sign Up' button, and a link to 'Sign In'. The illustration shows a credit card and a smartphone with the 'bBills' logo below them. A descriptive paragraph about the app is located at the bottom right.

**bBills**

Start analyzing your billing consumption

[Sign Up to bBills](#)

Email

Password

Sign Up

Have an account? [Sign In](#)

**bBills**

bBills is web App that enable you to add your bills, set your budget, manage your consumption, and save your money

**Figure 3.18: Welcome page**

## 2. HOME PAGE

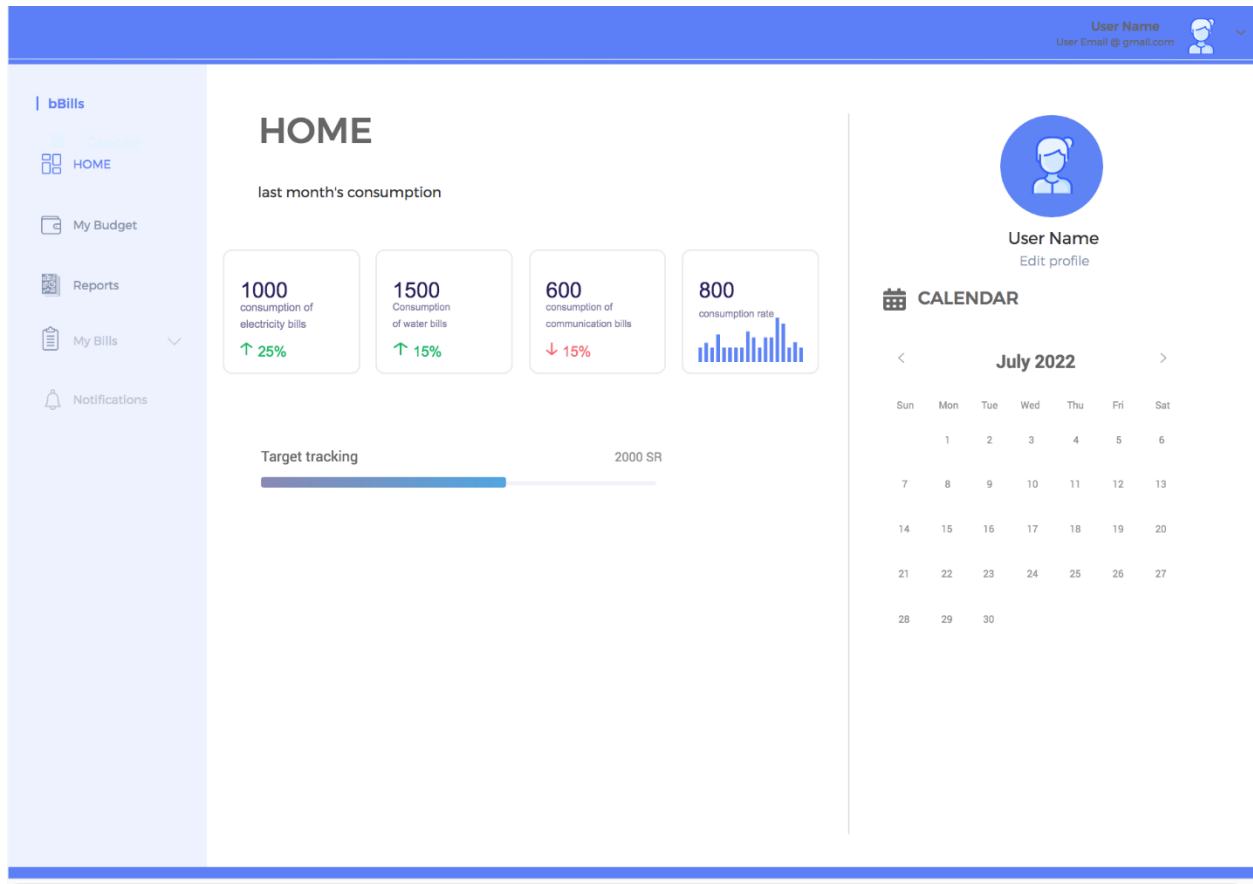


Figure 3.19: Home page

### 3. MY BUDGET PAGE

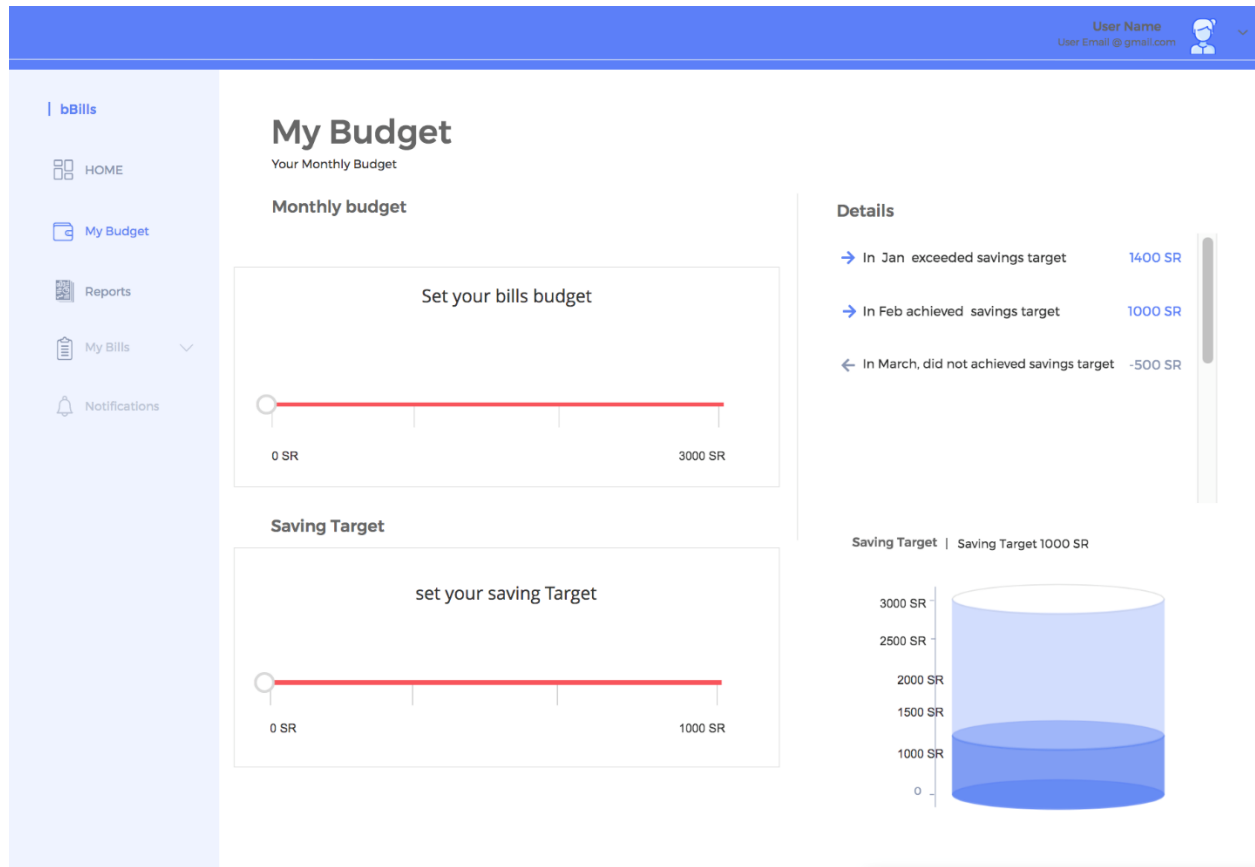


Figure 3.20: My budget page



## 4. MY BILLS PAGE

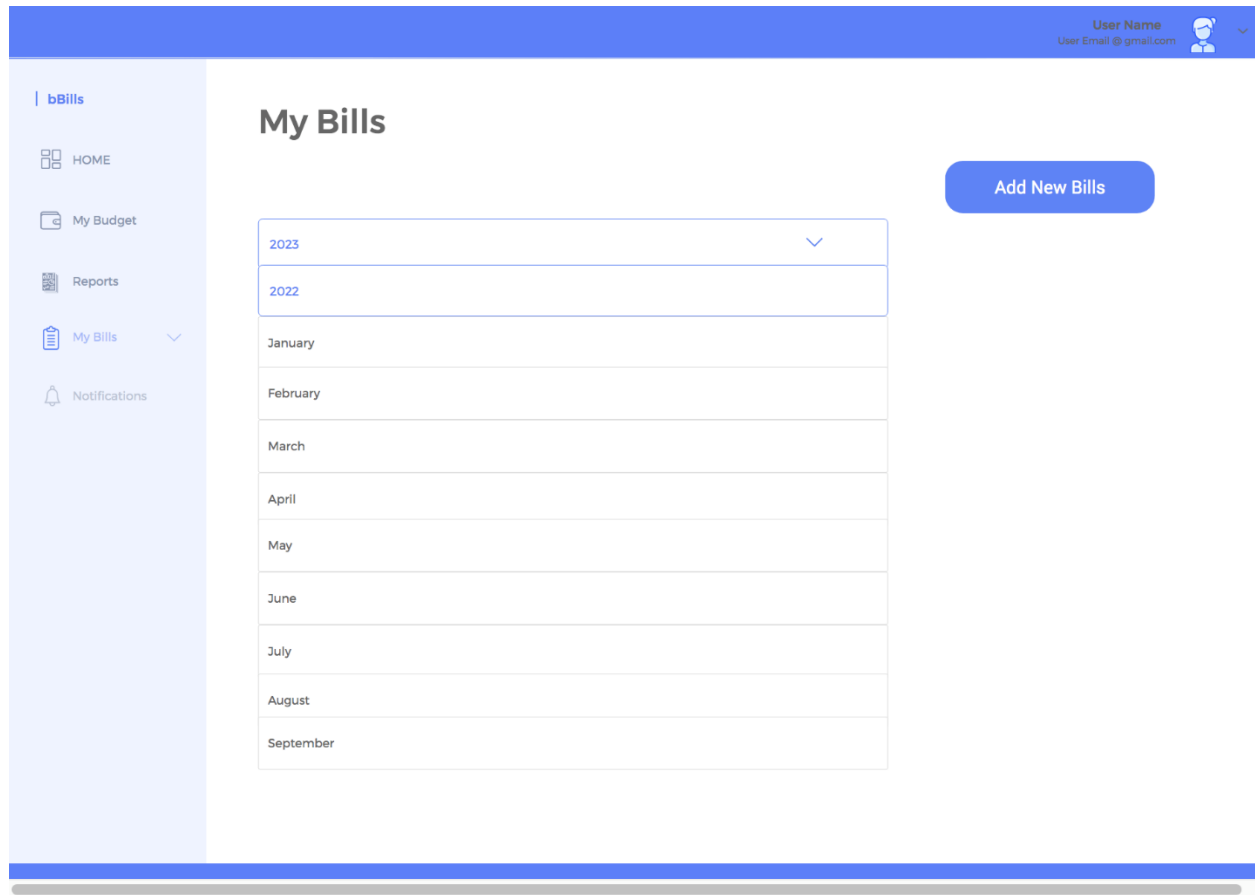


Figure 3.21: My bills page

## 5. ADD NEW BILLS PAGE

The screenshot displays the 'Add New Bills' page. At the top, a blue header bar contains the text 'User Name' and 'User Email @ gmail.com' next to a user icon and a dropdown arrow. On the left, a light blue sidebar lists navigation options: 'bBills', 'HOME', 'My Budget', 'Reports', 'My Bills' (which is selected and has a dropdown arrow), and 'Notifications'. The main content area is titled 'Add New Bill' and includes the instruction 'Fill your bill information'. Below this, there is a form with five input fields: a dropdown menu labeled 'select bill type', a text field for 'Bill Number', a date field containing '02/01/2023', a text field for 'Price', and a text field for 'Monthly Budget'. A blue 'Save' button is positioned below the form fields. The page is framed by a blue top bar and a blue bottom bar, with a grey horizontal line at the very bottom.

Figure 3.22: Add New Bills page

## 6. REPORT PAGE

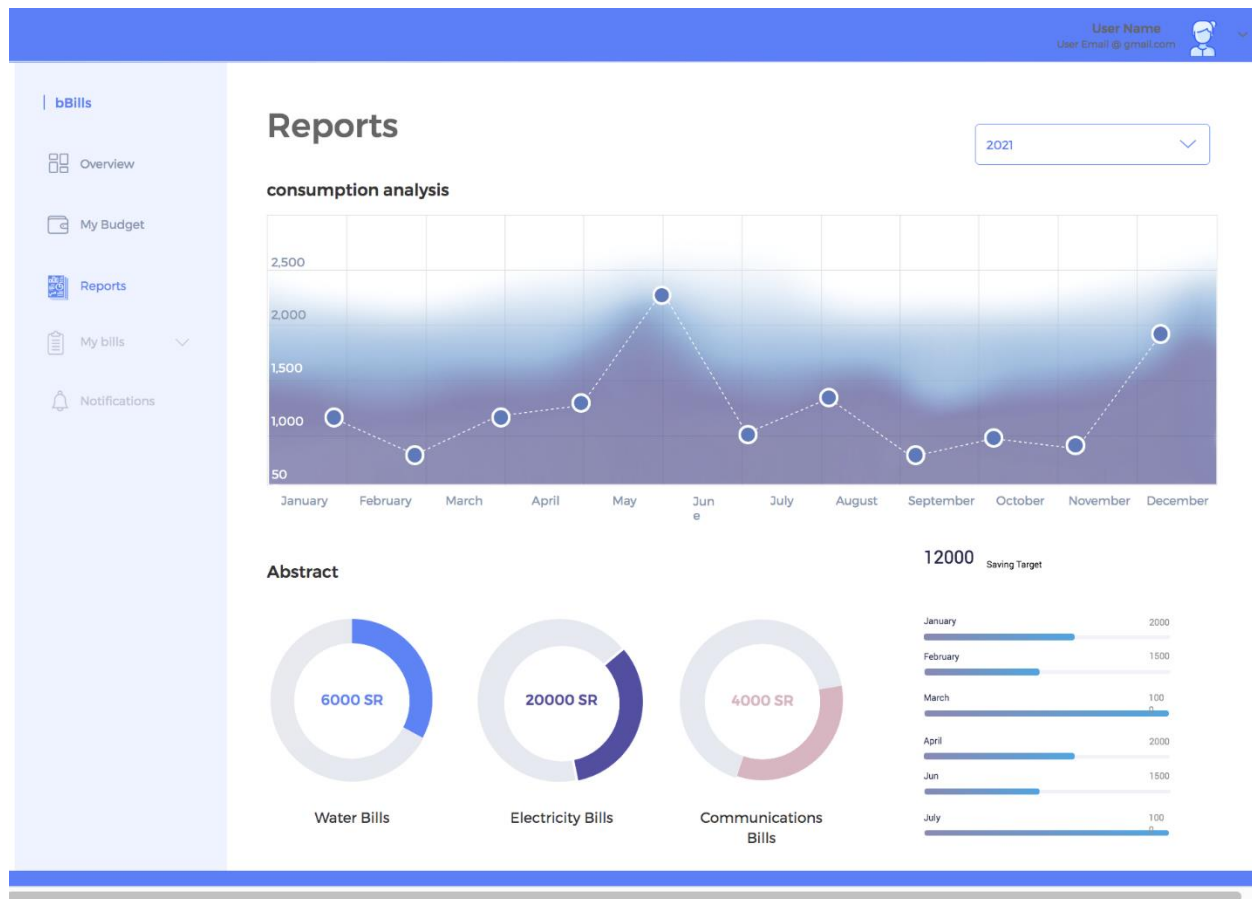


Figure 3.23: Report page

### 3.8. UNDERLYING TECHNIQUES

The system is responsible for analyzing the data using the Python programming language that has a wide range of libraries for numerical computation and data manipulation. In addition, the following methodology:

We built an artificial dataset based on a Gaussian normal distribution.

By using python programming language, we create four linear regression models, one for each type of bills.

The model has learned patterns and relationships from the training data, and it uses this knowledge to make predictions about the outcomes of new data.

We made the result available to the user through visual representations of data using the PHP programming language.

### 3.9. SUMMARY

The analysis phase is done by providing complete UML diagrams, this will show what to expect from the system and how it will be implemented and functioning. In this chapter we have also covered database and simple interface design.

## **4. CHAPTER 4 SYSTEM IMPLEMENTATION AND RESULTS**

### **4.1. INTRODUCTION**

System implementation is the phase where the designed system is put into action, and its effectiveness is evaluated. It involves activities such as coding, testing, deployment, and training to ensure the system meets the user requirements and is ready for use [4].

During system implementation, careful attention is given to data migration, integration with existing systems, and user acceptance testing, to ensure a smooth transition and successful adoption of the new system into the organization's operations.

### **4.2. INTERFACE DESIGN AND CODING**

The goal of the interface design and coding segment is to develop a simple and straightforward user interface for the system. In order to improve user experience, this requires developing the visual layout, navigational flow, and interactive features. Coding is also essential for implementing the system's features and business logic, turning the design into functional software components.

#### **4.2.1. BILLS DATASET**

Due to the lack of a data set regarding invoices in the Kingdom of Saudi Arabia, we had to create the data set artificially. Generating artificial data points that adhere to the statistical characteristics of a Gaussian distribution [5] is necessary to create an artificial dataset based on a Gaussian normal distribution. In order to do this, values from the distribution can be randomly selected, with the mean and standard deviation serving as the central tendency and spread, respectively, of the resulting data. The final dataset can be utilized for a variety of applications, including testing algorithms, running simulations, or enhancing real-world datasets for data analysis and machine learning tasks.

We have four types of target bills, electricity, phone (communication), water, and gas, first we need to determine the independent variable(s) for each type of these bills, the dependent variable for each bill is same which is bill' cost (the money amount of pay). The following table

illustrates the bills types and their independent variable(s), we consider the standard deviation of each distribution is 10%, with total samples 2400 for each dataset:

**Table 4.1: Datasets independent variables**

<b>Bill</b>	<b>Independent variables</b>	
Electricity	Living place type [apartment, villa, commercial]	Season [summer, spring, autumn, winter]
Phone	Internet type [DSL, 4G/5G, Fiber]	
Water	Bill duration [monthly, yearly]	
Gas	Bill duration [monthly, yearly]	

#### 4.2.1.1. Electricity bill

The following table views the average value of electricity bill cost (mean) depending on different values of living place type and season independent variables, where each combination is a Gaussian normal distribution:

**Table 4.2: Electricity dataset Gaussian distribution**

<b>Living type</b>	<b>Season</b>	<b>Cost average (Mean)</b>	<b>No of samples</b>
Apartment	Summer	500	200
Apartment	Spring	400	200
Apartment	Autumn	300	200
Apartment	Winter	200	200
Villa	Summer	950	200
Villa	Spring	850	200
Villa	Autumn	750	200
Villa	Winter	650	200
Commercial	Summer	1500	200
Commercial	Spring	1300	200
Commercial	Autumn	1100	200
Commercial	Winter	900	200

The following code creates the electricity bill dataset as data frame in Python programming language in jupyter notebook framework:

```
In [8]: 1 # Generate random cost values for each living type and season
2 #Loop for 12 iterations
3 for livingItem in living:
4     for season in seasons:
5         #print(avg[avg_row][avg_col])
6         avg_record = avg[avg_row][avg_col] # Get average cost value
7         std_value = avg_record*std/100 # Calculate standard deviation based on the average
8         x = random.normal(loc=avg_record, scale=std_value, size=(no_samples)) # Generate random samples
9         for xx in range(no_samples): # for example this loop generates 200 samples of apartment in summer and so on, apartme
10             cost_data.append(int(x[xx])) # Append cost value to the list
11             living_data.append(livingItem) # Append living type to the list
12             seasons_data.append(season) # Append season to the list
13             avg_col += 1 # now move from summer to spring and so on, keep sampe living type until end of all seasons (summer=>sp
14             avg_col = 0 # reset season to start (summer)
15             avg_row += 1 # move to next living [apartment=>villa=>comercial]
```

```
In [9]: 1 # Create a DataFrame from the generated data
2 df = pd.DataFrame({"living":living_data,
3                   "season":seasons_data,
4                   "cost":cost_data})
```

The following code views the generated dataset structure and then save the data frame as CSV file named “electric.csv”.

```
In [10]: 1 # view data frame
2 df
```

Out[10]:

	living	season	cost
0	apartment	summer	549
1	apartment	summer	445
2	apartment	summer	490
3	apartment	summer	533
4	apartment	summer	551
...	...	...	...
2395	comercial	winter	1073
2396	comercial	winter	980
2397	comercial	winter	1010
2398	comercial	winter	957
2399	comercial	winter	892

2400 rows × 3 columns

```
In [11]: 1 # Save the DataFrame to a CSV file
2 df.to_csv('electric.csv')
```

#### 4.2.1.2. Phone bill

The following table views the average value of phone bill cost (mean) depending on different values of internet type independent variable, where each combination is a Gaussian normal distribution:

**Table 4.3: Phone dataset Gaussian distributions**

Internet type	Cost average (Mean)	No of samples
DSL	120	800
5G/4G	150	800
Fiber	250	800

The following code creates the phone bill dataset as data frame in Python programming language in jupyter notebook framework:

```
In [6]: 1 for internet in internets:
2     avg_record = avg[avg_idx]
3     std_value = avg_record*std/100
4     x = random.normal(loc=avg_record, scale=std_value, size=(no_samples))
5     for xx in range(no_samples):
6         cost_data.append(int(x[xx]))
7         internet_data.append(internet)
8     avg_idx +=1
```

```
In [7]: 1 df = pd.DataFrame({"internet":internet_data,
2                        "cost":cost_data})
```

The following code views the generated dataset structure and then save the data frame as CSV file named “phone.csv”.



```
In [8]: 1 df
```

```
Out[8]:
```

	internet	cost
0	5G/4G	161
1	5G/4G	141
2	5G/4G	144
3	5G/4G	135
4	5G/4G	152
...	...	...
2395	DSL	119
2396	DSL	128
2397	DSL	139
2398	DSL	122
2399	DSL	112

2400 rows × 2 columns

```
In [8]: 1 df.to_csv('phone.csv')
```

#### 4.2.1.3. Water bill

The following table views the average value of water bill cost (mean) depending on different values of duration independent variable, where each combination is a Gaussian normal distribution:

**Table 4.4: Water dataset Gaussian distributions**

<b>Duration</b>	<b>Cost average (Mean)</b>	<b>No of samples</b>
Month	150	1200
Year	1600	1200

The following code creates the water bill dataset as data frame in Python programming language in jupyter notebook framework:

```
In [6]: 1 for duration in durations:
2         avg_record = avg[avg_idx]
3         std_value = avg_record*std/100
4         x = random.normal(loc=avg_record, scale=std_value, size=(no_samples))
5         for xx in range(no_samples):
6             cost_data.append(int(x[xx]))
7             duration_data.append(duration)
8         avg_idx +=1
```

```
In [7]: 1 df = pd.DataFrame({"duration":duration_data,
2                           "cost":cost_data})
```

The following code views the generated dataset structure and then save the data frame as CSV file named “water.csv”.

```
In [8]: 1 df
```

```
Out[8]:
```

	duration	cost
0	month	135
1	month	157
2	month	151
3	month	151
4	month	146
...	...	...
2395	year	1604
2396	year	1550
2397	year	1424
2398	year	1747
2399	year	1279

2400 rows × 2 columns

```
In [9]: 1 df.to_csv('water.csv')
```

#### 4.2.1.4. Gas bill

The following table views the average value of water bill cost (mean) depending on different values of duration independent variable, where each combination is a Gaussian normal distribution:

**Table 4.5: Gas dataset Gaussian distributions**

Duration	Cost average (Mean)	No of samples
Month	50	1200
Year	600	1200

The following code creates the water bill dataset as data frame in Python programming language in jupyter notebook framework:

```
In [4]: 1 durations = ["month", "year"]

In [5]: 1 std = 10 # Standard deviation percentage for generating random cost values
2 no_samples = 1200 # Number of samples to generate for each living type and season (1200 for month, 1200 for year)
3 duration_data = [] # List to store duration data
4 cost_data = [] # List to store cost data
5 avg_idx = 0

In [6]: 1 # Generate random cost values for each duration (month=>year)
2 # Loop of two iterations only
3 for duration in durations:
4     avg_record = avg[avg_idx] # (50=>600)
5     std_value = avg_record*std/100 # standard deviation value
6     x = random.normal(loc=avg_record, scale=std_value, size=(no_samples))
7     for xx in range(no_samples):
8         cost_data.append(int(x[xx]))
9         duration_data.append(duration)
10    avg_idx +=1 # move (50=>600)

In [7]: 1 df = pd.DataFrame({"duration":duration_data,
2                             "cost":cost_data})
```

The following code views the generated dataset structure and then save the data frame as CSV file named “gas.csv”.

```
In [8]: 1 df
```

```
Out[8]:
```

	duration	cost
0	month	49
1	month	57
2	month	54
3	month	51
4	month	52
...	...	...
2395	year	573
2396	year	613
2397	year	578
2398	year	631
2399	year	610

2400 rows × 2 columns

```
In [8]: 1 df.to_csv('gas.csv')
```

#### 4.2.2. LINEAR REGRESSION MODELS

One of the most important contributions of this project is the prediction of the appropriate invoice or bill value using artificial intelligent, we created four linear regression models, one for each type of bills.

Linear regression is a statistical modeling technique used to analyze the relationship between two variables: a dependent variable and one or more independent variables. It aims to find the linear relationship between the independent variables and the dependent variable [6].

In simple linear regression, there is only one independent variable, while multiple linear regression involves two or more independent variables. The dependent variable is often referred to as the target variable or the response variable, while the independent variables are called predictors or features.

The goal of linear regression is to fit a straight line or hyperplane to the data that minimizes the sum of the squared differences between the predicted values and the actual values of the

dependent variable. This line or hyperplane can then be used to make predictions about the dependent variable based on the values of the independent variables.

The equation for a simple linear regression can be represented as:

$$y = b_0 + b_1 * x$$

where:

- $y$  is the dependent variable
- $x$  is the independent variable
- $b_0$  is the y-intercept (the value of  $y$  when  $x$  is 0)
- $b_1$  is the slope of the line (the change in  $y$  for a unit change in  $x$ )

The values of  $b_0$  and  $b_1$  are estimated using various statistical techniques, such as the method of least squares, to find the best-fitting line to the data.

Linear regression has many applications in various fields, including economics, finance, social sciences, and machine learning. It provides insights into the relationship between variables, helps in predicting future values, and can be used for understanding the impact of independent variables on the dependent variable.

In this project, we use linear regression model from linear model of Scikit-learn. Scikit-learn, often abbreviated as `sklearn`, is a popular open-source machine learning library in Python. It provides a wide range of tools and algorithms for various machine learning tasks, such as classification, regression, clustering, dimensionality reduction, and model selection.

Scikit-learn is built on top of other scientific Python libraries, such as NumPy, SciPy, and matplotlib, and it integrates well with the broader Python ecosystem. It offers a consistent and user-friendly interface for implementing machine learning models and conducting data analysis.

Sklearn provides a consistent API, making it easy to switch between different algorithms and experiment with various approaches. It also offers utilities for data preprocessing, feature selection, and data transformation.

To store out trained models we used pickle library, In Python, the pickle module is used for object serialization and deserialization. Serialization refers to the process of converting objects into a byte stream, which can be stored in a file or transmitted across a network. Deserialization is the reverse process of reconstructing objects from the serialized byte stream.

The pickle module allows you to save any Python object (e.g., variables, data structures, or even custom classes) to a file or byte stream, and later retrieve them back into memory. It provides a convenient way to store and retrieve complex data structures or objects without losing their internal state.

#### 4.2.2.1. ELECTRICITY REGRESSION MODEL

After loading the corresponding electric dataset, where we have two categories independent variables which are living type and season, we need to convert theses categorical variables to numerical variables by replacing each category value with a suitable numerical value.

```
In [5]: 1 # Convert categorical values in the 'living' column to numerical values
        2 df['living_num'] = df['living'].replace({"apartment":1, "villa":2, "comercial":3})
        3 # Convert categorical values in the 'season' column to numerical values
        4 df['season_num'] = df['season'].replace({"summer":1, "spring":2, "autumn":3, "winter":4})
```

```
In [6]: 1 # display Random 200 samples
        2
        3 df.sample(200)
```

```
Out[6]:
```

	Unnamed: 0	living	season	cost	living_num	season_num
1987	1987	comercial	spring	1277	3	2
2377	2377	comercial	winter	885	3	4
678	678	apartment	winter	185	1	4
2219	2219	comercial	winter	1023	3	4
258	258	apartment	spring	395	1	2
...	...	...	...	...	...	...
2132	2132	comercial	autumn	1131	3	3
1607	1607	comercial	summer	1392	3	1
1520	1520	villa	winter	695	2	4
1996	1996	comercial	spring	1028	3	2
613	613	apartment	winter	224	1	4

200 rows × 6 columns

After that, we copy only these numerical variables to train electric regression model with fitting model with 75% of datasets as training samples and test the model with remainder 25% samples, using all data in training causes many problems like overfitting [7].

```
In [8]: 1 # Display information about the DataFrame
        2 df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2400 entries, 0 to 2399
Data columns (total 2 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   living_num  2400 non-null   int64
1   season_num  2400 non-null   int64
dtypes: int64(2)
memory usage: 37.6 KB
```

```
In [9]: 1 # Split the data into training and testing sets (25% for test, 75% for training)
        2 X_train, X_test, y_train, y_test = train_test_split(df2, target, test_size = 0.25)
```

```
In [10]: 1 # Create a LinearRegression model and fit it to the training data
         2 regr = LinearRegression()
         3 regr.fit(X_train, y_train)
         4 # Calculate the R^2 score of the trained model on the test data
         5 print(regr.score(X_test, y_test))
```

```
0.9365357431444992
```

We get accuracy of 0.94 which is enough to build our intelligent system, we can save now our system with pickle library, the following code store the fitted and trained model as electric.pkl file, and load it again, also we print the generated parameters of linear regression equation, where we have two independent variables, so we get a coefficient array with two elements and we get intercept value.

After that we tested the loaded model with a sample, where the living type is villa and the season is the summer, the model predicted the value of bill as 991 which is close to the original mean of the Gaussian normal distribution which equals to 950.

```

In [13]: 1 # Save the trained model to a file using pickle
          2 pickle.dump(regr, open('electric.pkl', 'wb'))

In [14]: 1 # Load the saved model from the file
          2 pickled_model = pickle.load(open('electric.pkl', 'rb'))

In [15]: 1 # Print the intercept and coefficients of the loaded model
          2
          3 print('Intercept: ', pickled_model.intercept_)
          4 print('Coefficients array: ', pickled_model.coef_)

Intercept: [269.01041944]
Coefficients array: [[ 429.26933793 -136.16847641]]

In [16]: 1 # Make a prediction using the loaded model
          2 living = 2 # Villa
          3 season = 1 # Summer
          4 sample_web = np.array([living, season])
          5 sample_web = sample_web.reshape(1, -1)
          6 pickled_model.predict(sample_web)

Out[16]: array([[991.38061889]])

```

#### 4.2.2.2. PHONE REGRESSION MODEL

After loading the corresponding phone dataset, where we have one categorical independent variable which is internet type we need to convert this categorical variable to numerical variable by replacing each category value with a suitable numerical value.



```
In [5]: 1 # Convert categorical values in the 'internet' column to numerical values
        2
        3 df['internet_num'] = df['internet'].replace({"5G/4G":2, "Fiber":3, "DSL":1})
```

```
In [6]: 1 df.sample(200)
```

```
Out[6]:
```

	Unnamed: 0	internet	cost	internet_num
1895	1895	DSL	119	1
1185	1185	Fiber	269	3
1410	1410	Fiber	281	3
1746	1746	DSL	115	1
336	336	5G/4G	158	2
...	...	...	...	...
49	49	5G/4G	141	2
95	95	5G/4G	150	2
977	977	Fiber	274	3
1848	1848	DSL	123	1
469	469	5G/4G	137	2

200 rows × 4 columns

After that, we copy only these numerical variables to train electric regression model with fitting model with 75% of datasets as training samples and test the model with remainder 25% samples, using all data in training causes many problems like overfitting as previous electric model.

```
In [7]: 1 # only one dimension data we need to reshape it
        2 X = np.array(df['internet_num']).reshape(-1, 1)
        3 y = np.array(df['cost']).reshape(-1, 1)
```

```
In [8]: 1 X
```

```
Out[8]: array([[2],
               [2],
               [2],
               ...,
               [1],
               [1],
               [1]], dtype=int64)
```

```
In [9]: 1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25)
```

```
In [10]: 1 regr = LinearRegression()
         2 regr.fit(X_train, y_train)
         3 print(regr.score(X_test, y_test))
```

0.8183552494604205

The accuracy is moderate, we need to enhance it to be over 90%, so we use polynomial feature transformer [8], in machine learning, the Polynomial Feature Transformer is a technique used to generate polynomial features from an existing set of features. It is available in various libraries, including scikit-learn (sklearn) in Python.

Polynomial features are created by taking the original features and generating new features as combinations of them, up to a specified degree. For example, given a set of features  $[x_1, x_2]$ , the polynomial feature transformer can create new features like  $[x_1^2, x_1x_2, x_2^2]$  or even higher-degree combinations like  $[x_1^3, x_1^2x_2, x_1x_2^2, x_2^3]$ .

The transformation is achieved by applying mathematical operations like exponentiation and multiplication to the original features. This process can be useful in capturing nonlinear relationships between features and the target variable, allowing linear models to learn more complex patterns.

The polynomial feature transformer is particularly helpful when the relationship between features and the target variable appears to be nonlinear. By generating polynomial features, it can help improve the performance of linear regression models or other models that assume a linear relationship between features and the target.

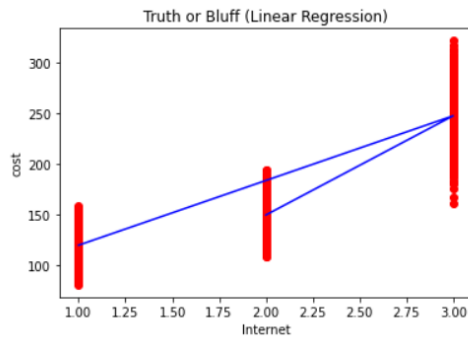
In the following code, polynomial regression is performed using the PolynomialFeatures transformer from scikit-learn. The input features  $X$  are transformed into polynomial features of degree 5 using `fit_transform`. Then, a linear regression model (LinearRegression) is fitted on the transformed features ( $X_{poly}$ ) and the target variable  $y$ .

The `viz_polynomial` function is defined to visualize the polynomial regression results. It creates a scatter plot of the original data points ( $X$  and  $y$ ) and plots the polynomial regression line by using `pol_reg.predict` on the transformed features. The resulting plot shows the relationship between the 'Internet' feature and the 'Cost' target variable

```

In [36]: 1
2 from sklearn.preprocessing import PolynomialFeatures
3 poly_reg = PolynomialFeatures(degree=5) # Create a polynomial feature transformer with degree 5
4 X_poly = poly_reg.fit_transform(X) # Transform the input features to include polynomial terms
5 pol_reg = LinearRegression() # Create a linear regression model
6 pol_reg.fit(X_poly, y) # Fit the linear regression model on the polynomial features
7
8 # Visualizing the Polynomial Regression results
9 def viz_polynomial():
10     plt.scatter(X, y, color='red') # Scatter plot of the original data points
11     plt.plot(X, pol_reg.predict(poly_reg.fit_transform(X)), color='blue') # Plot the polynomial regression line
12     plt.title('Truth or Bluff (Linear Regression)')
13     plt.xlabel('Internet')
14     plt.ylabel('cost')
15     plt.show()
16     return
17 viz_polynomial() # Call the visualization function

```



With the previous technique we enhanced accuracy by 10%, and get the target 90\$ of accuracy:

```

In [37]: 1 X_train_poly, X_test_poly, y_train_poly, y_test_poly = train_test_split(X_poly, y, test_size = 0.25)

In [38]: 1 # polynomial enhance model from 80% to 90 %
2 print(pol_reg.score(X_test_poly, y_test_poly))

0.9061571559936851

```

We can save now our system with pickle library, the following code store the fitted and trained model as phone.pkl file, and load it again, also we print the generated parameters of linear regression equation, where we have one original independent variable which will be transformed with polynomial degree of 5.

After that we tested the loaded model with a sample, where the internet type is fiber, the model predicted the value of bill as 247 which is close to the original mean of the Gaussian normal distribution which equals to 250.

```

In [39]: 1 pickle.dump(pol_reg, open('phone.pkl', 'wb'))

In [40]: 1 pickled_model = pickle.load(open('phone.pkl', 'rb'))

In [41]: 1 print('Intercept: ', pickled_model.intercept_)
2 print('Coefficients array: ', pickled_model.coef_)

Intercept: [1.72907858e+12]
Coefficients array: [[ 0.00000000e+00 -4.50179596e+11 -1.35341701e+12 -1.05868266e+12
 1.45050075e+12 -3.17300064e+11]]

In [42]: 1 from sklearn.preprocessing import PolynomialFeatures
2 poly_reg = PolynomialFeatures(degree=5)
3 internet = 3 # Fiber
4 sample_web = np.array([internet])
5 sample_web = sample_web.reshape(1, -1)
6 sample_web_poly = poly_reg.fit_transform(sample_web)
7 pickled_model.predict(sample_web_poly)
8

Out[42]: array([[247.79370117]])

In [43]: 1 predict = int(pickled_model.predict(sample_web_poly)[0][0])

In [44]: 1 predict

Out[44]: 247

```

#### 4.2.2.3. WATER REGRESSION MODEL

After loading the corresponding water dataset, where we have one categorical independent variable which is duration, we need to convert this categorical variable to numerical variable by replacing each category value with a suitable numerical value.

After that, we copy only these numerical variables to train electric regression model with fitting model with 75% of datasets as training samples and test the model with remainder 25% samples

```
In [5]: 1 df['duration_num'] = df['duration'].replace({"month":1, "year":2})
```

```
In [6]: 1 df.sample(200)
```

```
Out[6]:
```

	Unnamed: 0	duration	cost	duration_num
1568	1568	year	1599	2
561	561	month	138	1
464	464	month	158	1
1818	1818	year	1448	2
562	562	month	168	1
...	...	...	...	...
413	413	month	145	1
2227	2227	year	1547	2
979	979	month	157	1
1419	1419	year	1356	2
788	788	month	166	1

200 rows × 4 columns

```
In [7]: 1 X = np.array(df['duration_num']).reshape(-1, 1)
2 y = np.array(df['cost']).reshape(-1, 1)
```

```
In [8]: 1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25)
```

```
In [9]: 1 regr = LinearRegression()
2 regr.fit(X_train, y_train)
3 print(regr.score(X_test, y_test))
```

0.9750120129080768

We get accuracy of 0.975 which is enough to build our intelligent system, we can save now our system with pickle library, the following code store the fitted and trained model as water.pkl file, and load it again, also we print the generated parameters of linear regression equation, where we have one dependent variable, so we get a coefficient array with one element and we get intercept value as shown by the print method.

```

In [12]: 1 pickle.dump(regr, open('water.pkl', 'wb'))

In [13]: 1 pickled_model = pickle.load(open('water.pkl', 'rb'))

In [14]: 1 print('Intercept: ', pickled_model.intercept_)
          2 print('Coefficients array: ', pickled_model.coef_)

Intercept:  [-1306.37063835]
Coefficients array:  [[1455.46734165]]

In [15]: 1 duration = 2 # year
          2 sample_web = np.array([duration])
          3 sample_web = sample_web.reshape(1, -1)
          4 pickled_model.predict(sample_web)

Out[15]: array([[1604.56404494]])

```

The previous code loaded stored model, test a sample of duration year and predicted invoice value 1600 which is very close to the mean of distribution which is 1600.

#### 4.2.2.4. GAS REGRESSION MODEL

After loading the corresponding water dataset, where we have one categorical independent variable which is duration, we need to convert this categorical variable to numerical variable by replacing each category value with a suitable numerical value.

We get accuracy of 0.977 which is enough to build our intelligent system, we can save now our system with pickle library, the following code store the fitted and trained model as water.pkl file, and load it again, also we print the generated parameters of linear regression equation, where we have one dependent variable, so we get a coefficient array with one element and we get intercept value as shown by the print method.

```

In [5]: 1 df['duration_num'] = df['duration'].replace({"month":1, "year":2})

In [6]: 1 df.sample(200)

Out[6]:
   Unnamed: 0  duration  cost  duration_num
1250      1250    year   631              2
554        554   month    43              1
2225      2225    year   607              2
1697      1697    year   615              2
65         65   month    51              1
...
1421      1421    year   572              2
662        662   month    51              1
2138      2138    year   628              2
1303      1303    year   642              2
855        855   month    49              1

200 rows x 4 columns

In [7]: 1 X = np.array(df['duration_num']).reshape(-1, 1)
        2 y = np.array(df['cost']).reshape(-1, 1)

In [8]: 1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25)

In [9]: 1 regr = LinearRegression()
        2 regr.fit(X_train, y_train)
        3 print(regr.score(X_test, y_test))

0.9767486732084127

```

After that we tested the loaded model with a sample, where the duration value is year and, the model predicted the value of bill as 599 which is close to the original mean of the Gaussian normal distribution which equals to 600.

```

In [12]: 1 pickle.dump(regr, open('gas.pkl', 'wb'))

In [13]: 1 pickled_model = pickle.load(open('gas.pkl', 'rb'))

In [14]: 1 print('Intercept: ', pickled_model.intercept_)
          2 print('Coefficients array: ', pickled_model.coef_)

Intercept:  [-499.94235188]
Coefficients array:  [[549.6813845]]

In [15]: 1 duration = 2 # year
          2 sample_web = np.array([duration])
          3 sample_web = sample_web.reshape(1, -1)
          4 pickled_model.predict(sample_web)

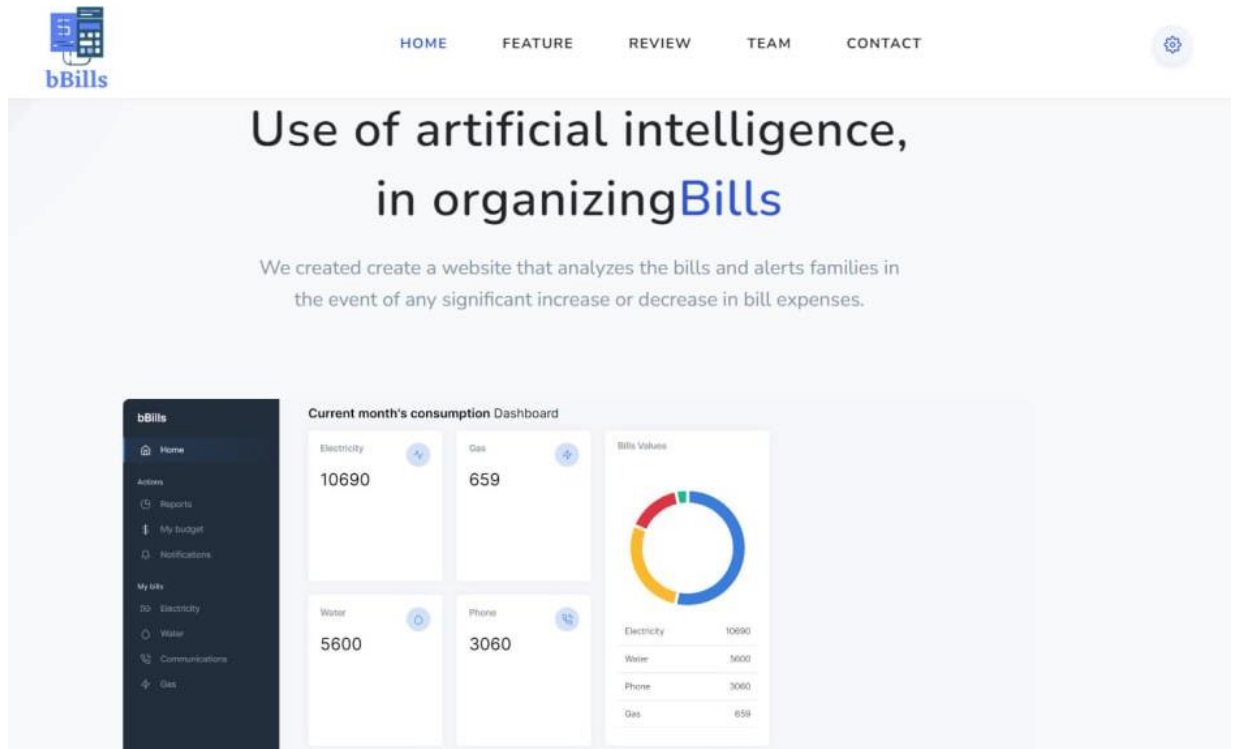
Out[15]: array([[599.42041712]])

```

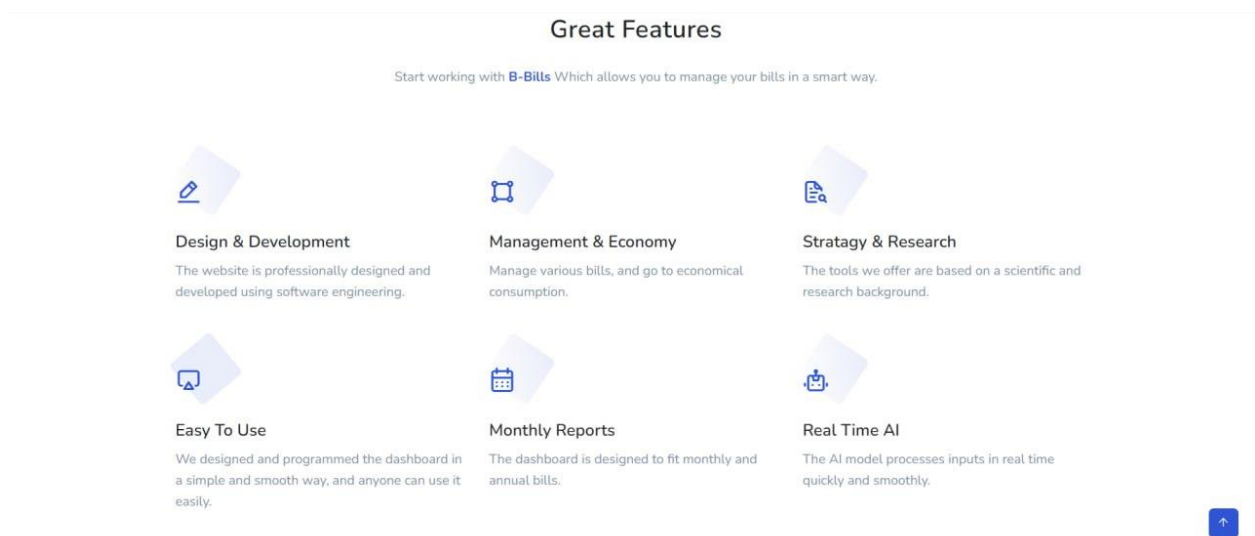
### 4.2.3. LANDING PAGE MODULE

We created index.php file as landing page without PHP or Python code, just static images and text, with CSS code and JS, the following screenshots illustrates the landing page design:

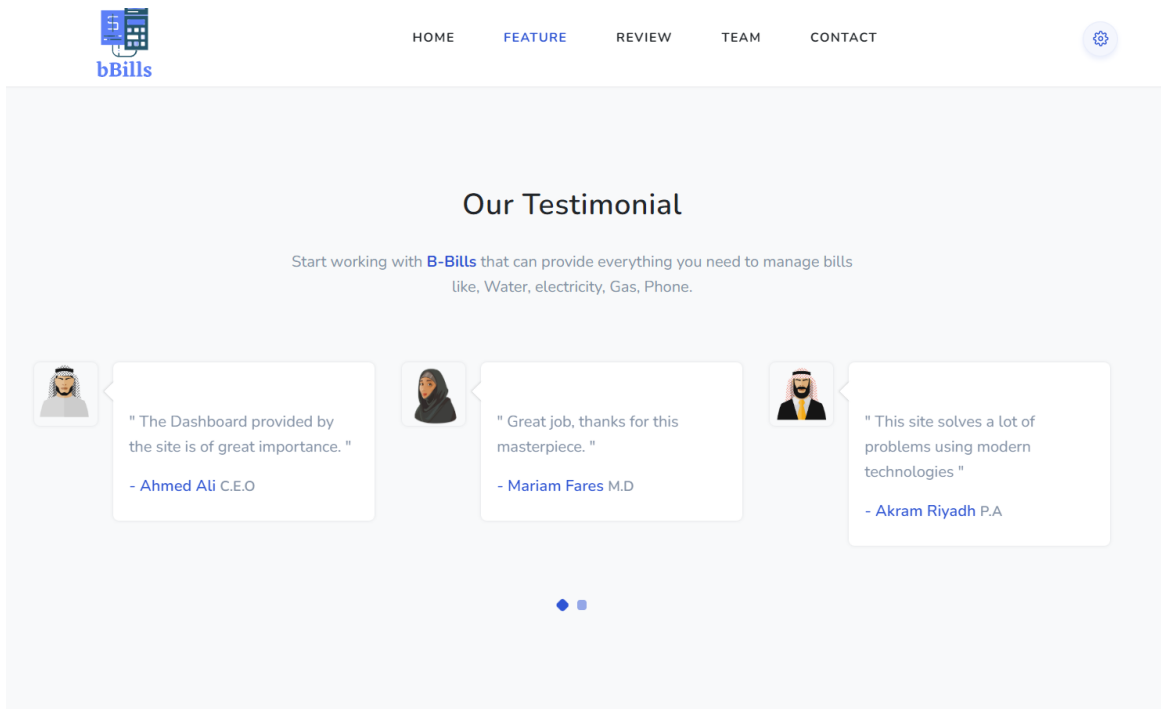




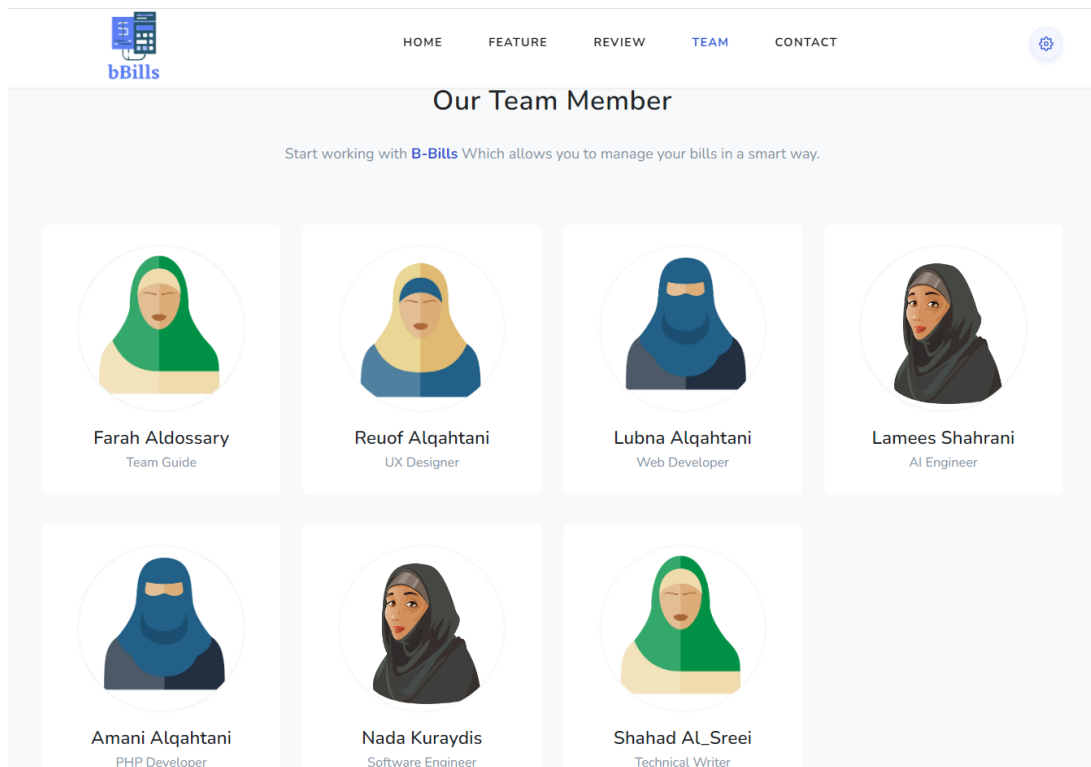
**Figure 4.1: Landing page hero section**




**Figure 4.2: Landing page features section**




**Figure 4.3: Landing page review section**



**Figure 4.4: Landing page our team section**



[HOME](#)
[FEATURE](#)
[REVIEW](#)
[TEAM](#)
[CONTACT](#)



## Get In Touch !

Start working with **B-bills** that can provide everything you need to generate awareness, drive traffic, connect.

**Your Name \***

First Name :

**Your Email \***

Your email :


**Comments**


Your Message :


[Send Message](#)

### Contact Details

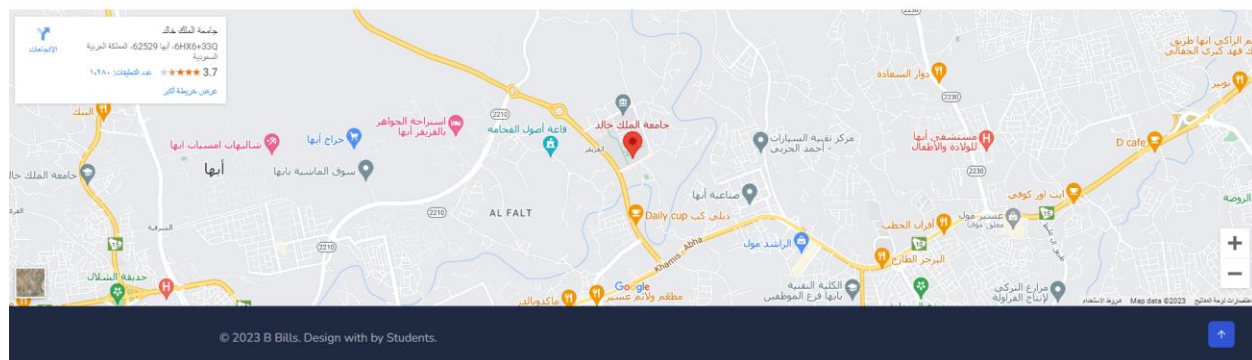
Start working with **B-Bills** We seek to benefit from technological development and spread it in the Kingdom of Saudi Arabia

 **Email**  
contact@bbills.sa.com

 **Phone**  
+966-54-123-4567

 **Location**  
[View on Google map](#)

**Figure 4.5: Landing page contact section**



**Figure 4.6: Landing page map and footer**

#### 4.2.4. DASHBOARD WEB APPLICATION DEVELOPMENT

Rather than the landing webpage we developed a dashboard for user to manage his bills using PHP as primary programming language with HTML5 and bootstrap framework to enhance the design, the AI module is written in Python Programming language, and executed using Common Programming Interface (CGI) component to allow running external scripts like Python in Apache server.

In developing the proposed dashboard we used Model-View-Controller pattern, MVC stands for Model-View-Controller, which is a software architectural pattern commonly used in the development of web applications and other software systems. It separates the application into three interconnected components: the model, the view, and the controller.

1. **Model:** The model represents the data and the business logic of the application. It encapsulates the data structures, database interactions, and rules for manipulating and processing the data.
2. **View:** The view is responsible for presenting the user interface and displaying the data to the users. It renders the information from the model into a format that can be easily understood by the users, such as HTML pages or user interface elements.
3. **Controller:** The controller acts as an intermediary between the model and the view. It receives the user's input and translates it into actions to be performed by the model or the view. It handles user interactions, updates the model based on user input, and updates the view to reflect changes in the model.

The MVC pattern promotes separation of concerns, making the code more modular, maintainable, and testable. It allows for independent development and modification of each component, enabling easier code reuse and flexibility in implementing different user interfaces or modifying the underlying data structures.

## **4.2.5. MODEL CLASSES MODULE**

### **4.2.5.1. USER MODEL CLASS**

Which is responsible of fetching data from the following database tables:

1. User
2. User\_budget
3. User\_targets
4. User\_details

The class has the following methods:

```
function checkEmail($db, $email)
```

to check if the email is available during registration process.

```
function checkPhone($db, $phone)
```

to check if the phone is available during registration process.

```
function getUserID($db, $email)
```

get user id from given email

```
function getUserProfile($db, $id)
```

get all user data from user table by its id

```
function getUserDetails($db, $id)
```

get user details from user\_details table by its id

```
function getUserBudget($db, $id)
```

get user budgets from user\_budget table by its id

```
function getUserTarget($db, $id)
```

get user targets from user\_targets table by its id

#### 4.2.5.2. BILL MODEL CLASS

Which is responsible of fetching data from the following database tables:

1. electricity\_bill
2. phone\_bill
3. water\_bill
4. gas\_bill

The class has the following methods:

```
function getElectricBillByID($db, $id)
```

get electric bill information from electricity\_bill table by electricity bill id

```
function getWaterBillByID($db, $id)
```

get water bill information from water\_bill table by water bill id

```
function getGasBillByID($db, $id)
```

get gas bill information from gas\_bill table by gas bill id

```
function getPhoneBillByID($db, $id)
```

get phone bill information from phone\_bill table by phone bill id

```
function getTotalElectricCost($db, $user_id)
```

get the sum of all electric bills cost by user id

```
function getTotalGasCost($db, $user_id)
```

get the sum of all gas bills cost by user id

```
function getTotalWaterCost($db, $user_id)
```

get the sum of all water bills cost by user id

```
function getTotalPhoneCost($db, $user_id)
```

get the sum of all phone bills cost by user id

```
function getElectricCostMonthYear($db, $user_id, $month, $year)
```

get the sum of all electric bills of given user id within given month and year

```
function getPhoneCostMonthYear($db, $user_id, $month, $year)
```

get the sum of all phone bills of given user id within given month and year

```
function getWaterCostMonthYear($db, $user_id, $month, $year)
```

get the sum of all water bills of given user id within given month and year

```
function getGasCostMonthYear($db, $user_id, $month, $year)
```

get the sum of all gas bills of given user id within given month and year

and many others functions to get information about user's bills.

## 4.2.6. CONTROLLER CLASSES MODULE

### 4.2.6.1. USER CONTROLLER CLASS

Which is responsible of inserting and updating data of the following database tables:

1. User
2. User\_details
3. User\_budget
4. User\_targets

And has the following methods:

```
function register($db,$postdata=array())
```

to register user and insert his submit data in user, user\_details, user\_budget, user\_targets tables.

```
function login($db, $postdata=array())
```

login function with given email and password

```
function updateBudget($db, $postdata)
```

update user's budgets

```
function updateTarget($db, $postdata)
```

update user's targets

#### 4.2.6.2. BILL CONTROLLER CLASS

Which is responsible of inserting and updating data of the following database tables:

1. Electricity\_bill
2. Water\_bill
3. Phone\_bill
4. Gas\_bill

And has the following methods:

```
function addElectricBill($db,$postdata=array())
```

add electric bill for given user id

```
function addWaterBill($db,$postdata=array())
```

add water bill for given user id

```
function addPhoneBill($db,$postdata=array())
```

add phone bill for given user id

```
function addGasBill($db,$postdata=array())
```

add gas bill for given user id

```
function addElectricResult($db, $bill_id, $result, $targetCode)
```

update electric bill information the electricity\_bill\_result and target\_result of given bill\_id

Table 4.6: Add bill result code

Code	electricity_bill_result	target_result
0	Not set yet	Not set yet
1	Within predicted range ( $\pm 10\%$ )	Achieved saving target
2	Over the predicted range $> + 10\%$	Not achieved saving target
3	Below the predicted range $< -10\%$	Exceeds saving target

The previous implementation is repeated for all bills types (water, phone, and gas)

#### 4.2.7. REGISTER MODULE

```
<?php
```

```

include('config.php');
include('models/user.php');
include('controllers/user.php');

$errormsg="";
$messagesucces = "";

if($_SERVER["REQUEST_METHOD"] == "POST") {

    $userModel= new UserModel();
    $emailavailable = $userModel->checkEmail($db, $_POST['email']);
    $phoneavailable = $userModel->checkPhone($db, $_POST['phone']);

    if(!$emailavailable){
        $errmsg="Your Email is already registered with us";
    }

    elseif(!$phoneavailable){
        $errmsg="Your phone is already registered with us";
    }
    else{
        $UserController = new UserController();
        $UserController->register($db, $_POST);
        $messagesucces ="successfully registered";
    }

}

?>

```



## Get started

To help families manage their expenses and pay their bills efficiently

Name

Phone

Email

Password

Living In

Internet type

Water Bill duration

Gas Bill duration

Figure 4.7: Register interface

### 4.2.8. LOGIN MODULE

```
<?php
```

```

include('config.php');
session_start();
include('models/user.php');
include('controllers/user.php');
$errormsg="";

if($_SERVER["REQUEST_METHOD"] == "POST") {

    $userModel = new UserModel();
    $userController =new UserController();
    $loginresult=$userController->login($db, $_POST);

    if($loginresult==0){
        $errmsg="Email is not existing !!!!";
    }

elseif($loginresult==2){
    $errmsg="Password incorrect !!!";
}

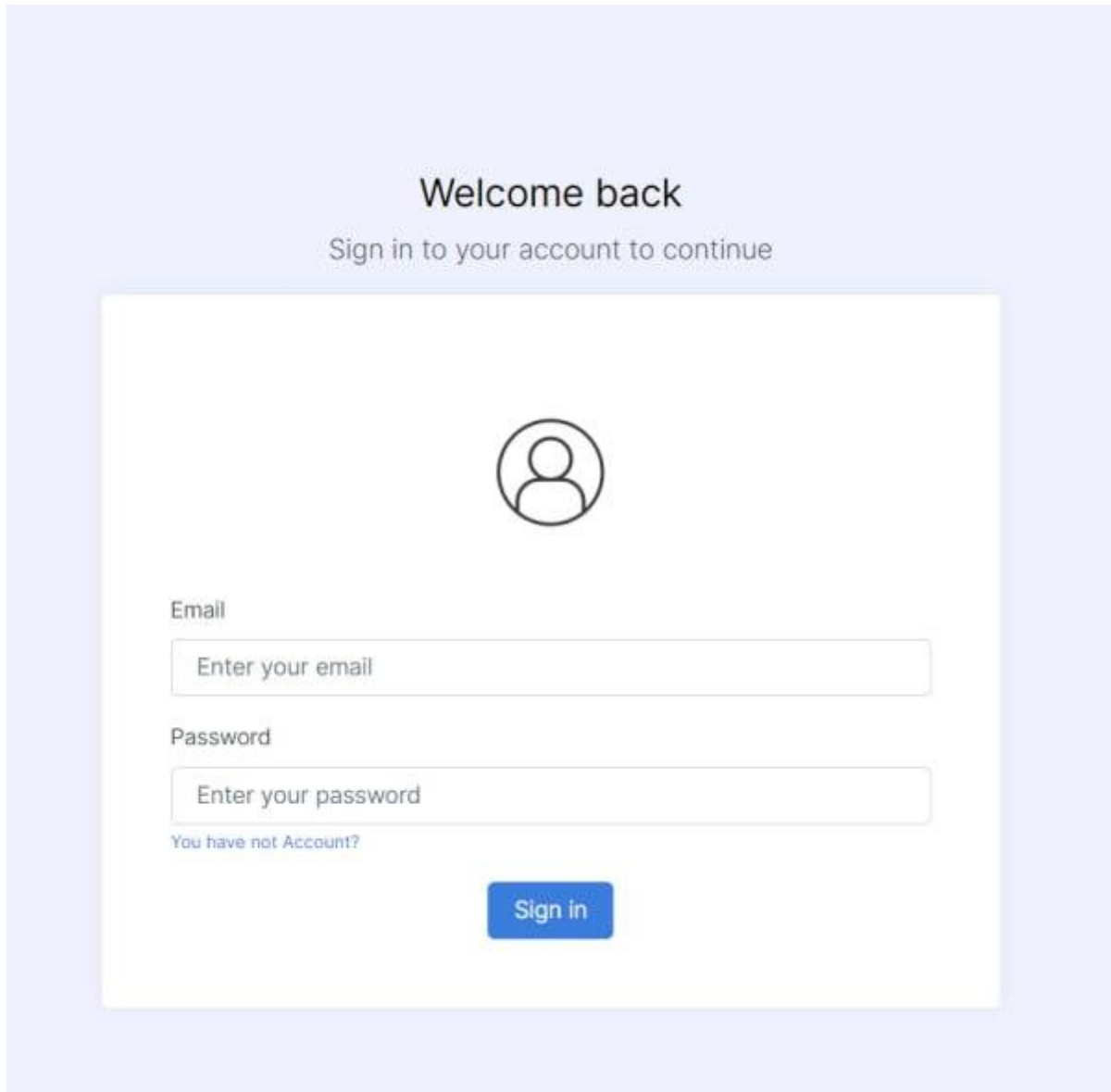
else{
    $errmsg="يتم التسجيل";

    $_SESSION['login_user'] = $_POST['email'];
    $_SESSION['user_id'] = $userModel->getUserID($db,$_POST['email'] );
    //echo $_SESSION['user_id'];
    header("location: dashboard.php");
}

}

?>

```



The image shows a login interface with a light blue background. At the top, the text "Welcome back" is centered in a bold, dark font, followed by "Sign in to your account to continue" in a smaller, lighter font. Below this is a white rectangular box with a subtle shadow. Inside the box, there is a circular icon of a person. Underneath the icon are two input fields: one for "Email" and one for "Password", both with placeholder text "Enter your email" and "Enter your password" respectively. Below the password field is a link that says "You have not Account?". At the bottom of the white box is a blue button with the text "Sign in" in white.

Figure 4.8: Login interface

#### 4.2.9. ADD BILL MODULE

```
<?php
include('session-user.php');

$errormsg="";
$messagesucces = "";
include("controllers/bill.php");
include('models/user.php');

if($_SERVER["REQUEST_METHOD"] == "POST") {
```

```
$billController = new BillController($user_id);  
$bill_id = $billController->addElectricBill($db, $_POST);  
  
header("location: electric.py?bill_id=".$bill_id);  
  
}  
  
$userModel = new UserModel();  
$budgetObj = $userModel->getUserBudget($db, $user_id);  
$targetObj = $userModel->getUserTarget($db, $user_id);  
  
//$electricBudget = $budgets->electricity;
```

## Add Electric Bill

Fill the following form

Bill Number:

Bill Date:



تکلیف - رهش - موي

Bill Amount (SR):

Budget (SR):

Add bill

Figure 4.9: Add bill interface

### 4.2.10. VIEW BILL MODULE

```
<?php
include('../config.php');
session_start();
date_default_timezone_set('Asia/Riyadh');
```

```

$todaydate=date('Y-m-d H:i:s');
$myusername=$_SESSION['login_user'];
$myid=$_SESSION['user_id'];

$sql = "SELECT * from electricity_bill where user_id='$myid'";

//echo $sql;
$result = mysqli_query($db,$sql);
$array=array();
while($row = mysqli_fetch_assoc($result)) {
    $array[] = $row;
}

$dataset = array(
    "echo" => 1,
    "totalrecords" => count($array),
    "totaldisplayrecords" => count($array),
    "data" => $array
);

echo json_encode($dataset);
?>

```

#### MY BILLS (Electricity)

Add new bill

Show 10

Search Data...

Bill No	Bill Date	Actual Cost	Predicted Cost	Created At	Delete
78961234011	2022-11-01	750	721	2023-05-10 02:10:56	
78961234021	2022-01-01	650	588	2023-05-10 02:10:56	
78961234022	2022-02-01	600	588	2023-05-10 02:10:56	
78961234023	2022-03-01	800	588	2023-05-10 02:10:56	
78961234024	2022-04-01	700	855	2023-05-10 02:10:56	
78961234025	2022-05-01	900	855	2023-05-10 02:10:56	
78961234026	2022-06-01	1150	855	2023-05-10 02:10:56	
78961234027	2022-07-01	1000	989	2023-05-10 02:10:56	
78961234028	2022-08-01	950	989	2023-05-10 02:10:56	
78961234029	2022-09-02	900	989	2023-05-10 02:10:56	

Showing 1 to 10 of 13 entries

Previous

1

2

Next

**Figure 4.10: View bill interface**

#### 4.2.11. VIEW BILL MODULE

```
<?php
include('session-user.php');
include('models/bill.php');
include('controllers/bill.php');
include('models/codemap.php');

$bill_id = $_GET['id'];

$billModel = new BillModel();

$billOBJ = $billModel->getElectricBillByID($db, $bill_id);

$diff = $billOBJ->bill_cost - $billOBJ->predicted_cost;

$diffPercentage = round(($diff/$billOBJ->predicted_cost*100), 2);

$time_input = strtotime($billOBJ->bill_date);
$date_input = getDate($time_input);

$billmonth = $date_input['month'];

$billyear = $date_input['year'];

$billController = new BillController($user_id);

$resultCode = 1;

if ($diffPercentage>10){
    $resultCode = 2;
}
else if($diffPercentage<-10){

    $resultCode = 3;
}

$targetCode = 1;

$actualSave = $billOBJ->electricity_budget - $billOBJ->bill_cost;

if($actualSave > $billOBJ->electricity_target){
```

```
        $targetCode = 3;
    }

    if($actualSave < $billOBJ->electricity_target){

        $targetCode = 2;
    }

    $billController->addElectricResult($db, $bill_id, $resultCode, $targetCode);

    $codeModel = new CodeModel();

    $resultTXT = $codeModel->resultTXT($resultCode);

    ?>
```



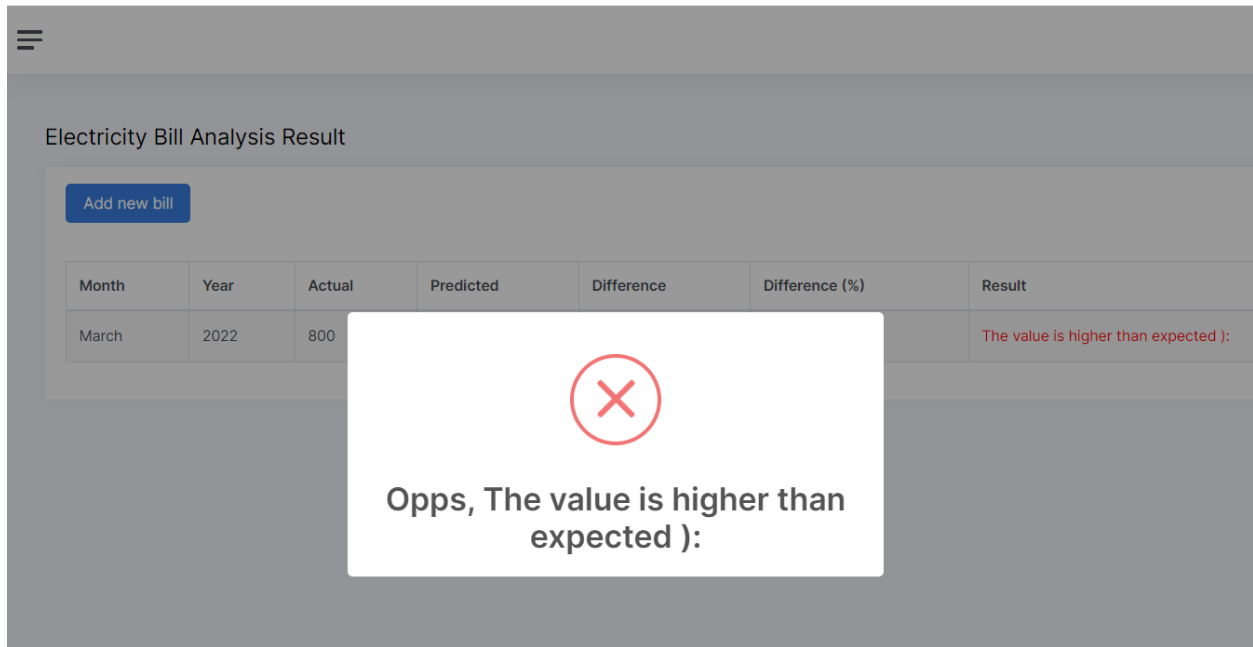


Figure 4.11: Danger notification interface

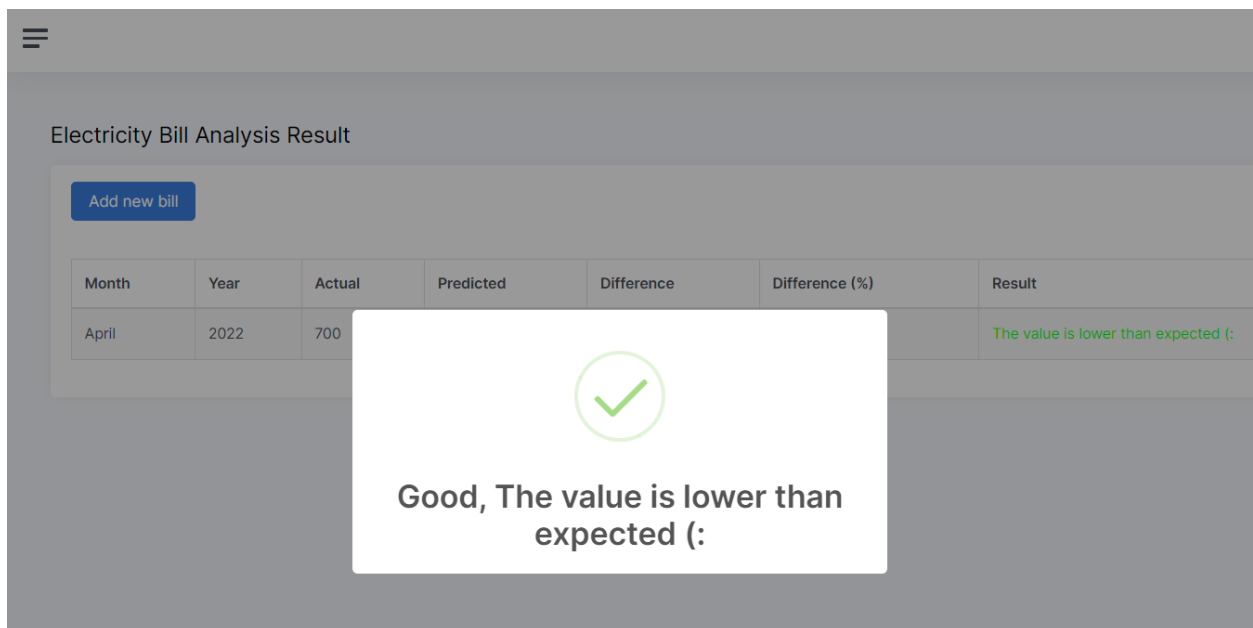


Figure 4.12: Scuccess notification interface

#### 4.2.12.DASHBOARD home-page module

```
<?php
include('session-user.php');
```

```

include('models/bill.php');
include('models/user.php');

$billModel = new BillModel();
$electricTotal = $billModel->getTotalElectricCost($db, $user_id);
$gasTotal = $billModel->getTotalGasCost($db, $user_id);
$waterTotal = $billModel->getTotalWaterCost($db, $user_id);
$phoneTotal = $billModel->getTotalPhoneCost($db, $user_id);

$userModel = new UserModel();
$userDetailsObj = $userModel->getUserDetails($db, $user_id);

$water_duration = $userDetailsObj->water_duration;
$gas_duration = $userDetailsObj->gas_duration;

$year = '2022';
?>

```

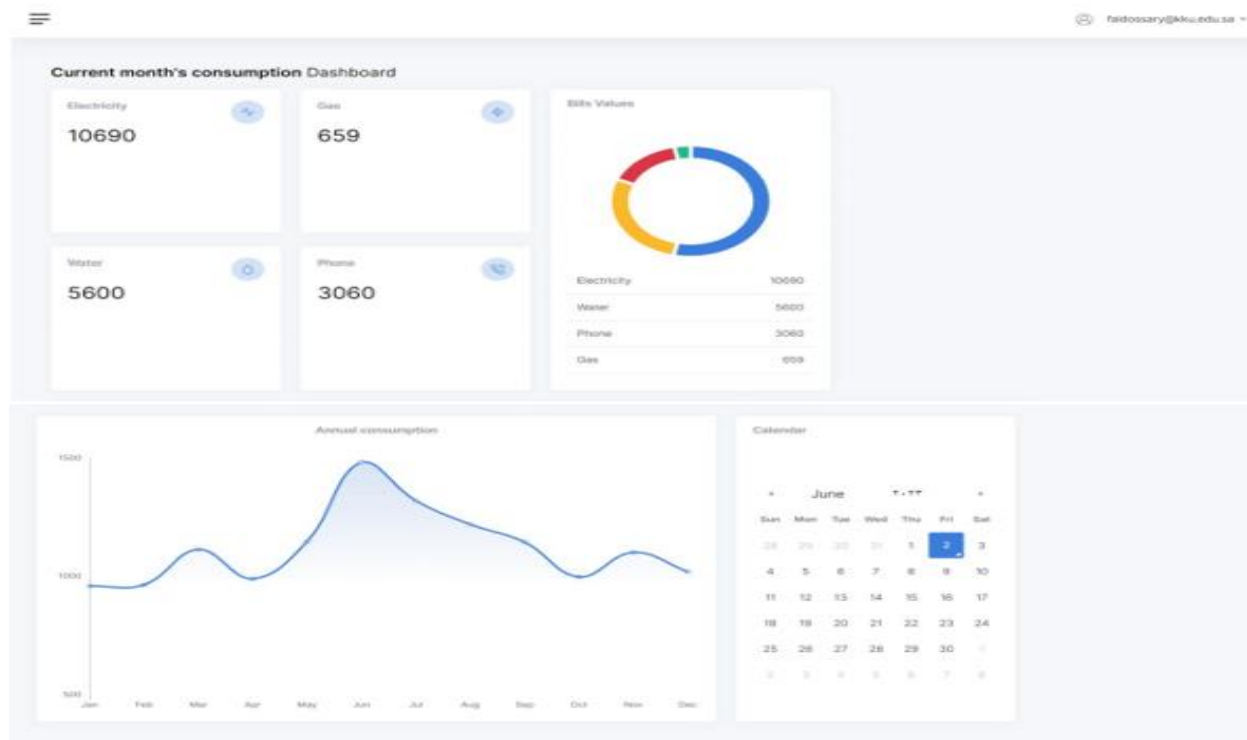


Figure 4.13: Main dashboard interface

## 4.2.13.BUDGET module

```
<?php
```

```

include('session-user.php');
include('models/user.php');
include('controllers/user.php');
include('models/bill.php');

$msg = "";

if($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['update-budget'])) {

    $userController =new UserController();
    $userController->updateBudget($db, $_POST);

    $msg = "Budget was updated";
}

if($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['update-target'])) {

    $userController =new UserController();
    $userController->updateTarget($db, $_POST);

    $msg = "Budget was updated";
}

$billModel = new BillModel();

$electricSave = $billModel->electricSave($db, $user_id);
$waterSave = $billModel->waterSave($db, $user_id);
$phoneSave = $billModel->phoneSave($db, $user_id);
$gasSave = $billModel->gasSave($db, $user_id);
$userModel = new UserModel();
$budgetObj = $userModel->getUserBudget($db, $user_id);
$targetObj = $userModel->getUserTarget($db, $user_id);

$userDetails = $userModel->getUserDetails($db, $user_id);

include('models/targets.php');

$targetModel = new TargetsModel();

$electrics = $targetModel->getTargetsElectric($db, $user_id);
$phones = $targetModel->getTargetsPhone($db, $user_id);
$waters = $targetModel->getTargetsWater($db, $user_id);

```

```

$gases = $targetModel->getTargetsGas($db, $user_id);
include('models/codemap.php');

$codeMapModel =new CodeModel();

?>

```

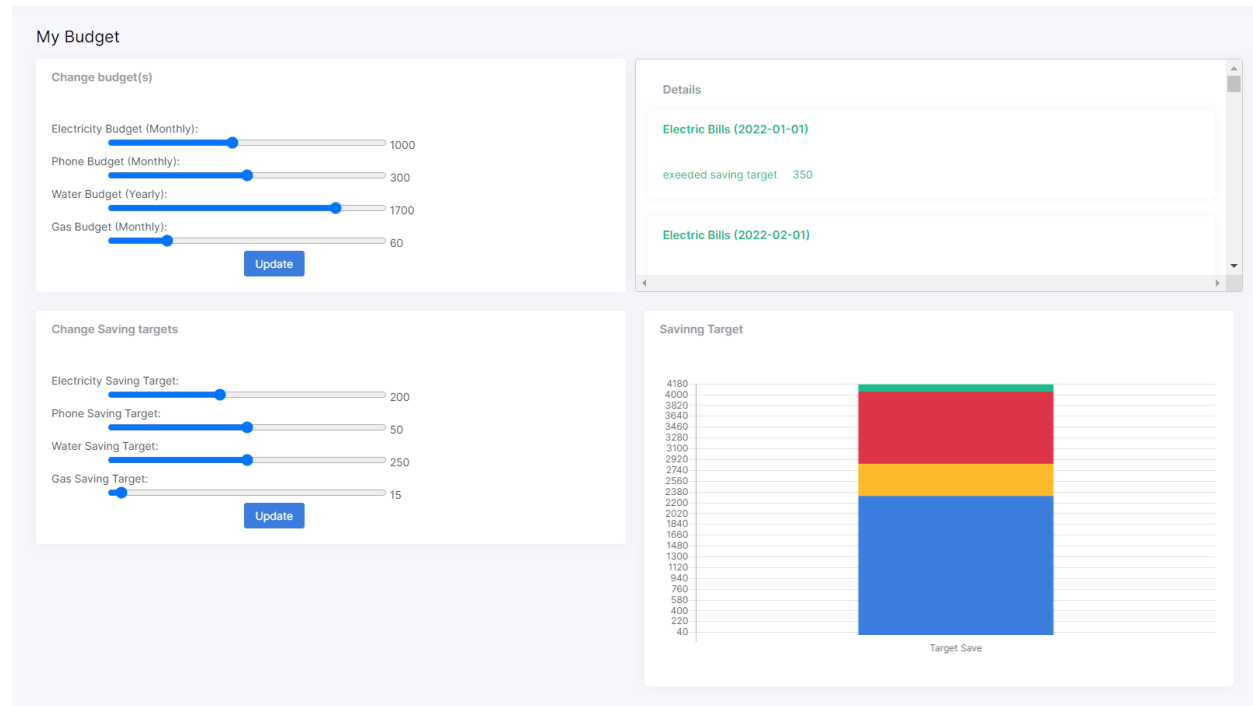


Figure 4.14: Budget interface

## 4.2.14.REPORT MODULE

```

<?php
include('session-user.php');

include('models/bill.php');
include('models/user.php');

$billModel = new BillModel();
$electricTotal = $billModel->getTotalElectricCost($db, $user_id);
$gasTotal = $billModel->getTotalGasCost($db, $user_id);
$waterTotal = $billModel->getTotalWaterCost($db, $user_id);
$phoneTotal = $billModel->getTotalPhoneCost($db, $user_id);

$userModel = new UserModel();
$userDetailsObj = $userModel->getUserDetails($db, $user_id);

```

```

$water_duration = $userDetailsObj->water_duration;
$gas_duration = $userDetailsObj->gas_duration;

$year = '2022';

?>

```



Figure 4.15: Reports interface

## 4.2.15.NOTIFICATIONS MODULE

```

<?php

include('session-user.php');
include('models/notifications.php');
include('models/codemap.php');

$notificationModel = new NotificationsModel();
$electrics = $notificationModel->getElectricNotifications($db, $user_id);
$waters = $notificationModel->getWaterNotifications($db, $user_id);
$gases = $notificationModel->getGasNotifications($db, $user_id);

```

```

$phones = $notificationModel->getPhoneNotifications($db, $user_id);
//print_r($electric);
$codeMapModel =new CodeModel();

?>

```

Notifications	
<p><b>Electric Bills (2022-01-01)</b></p> <p>The value is higher than expected ):</p> <p>Show Results</p>	<p><b>Electric Bills (2022-03-01)</b></p> <p>The value is higher than expected ):</p> <p>Show Results</p>
<p><b>Electric Bills (2022-04-01)</b></p> <p>The value is lower than expected (:</p> <p>Show Results</p>	<p><b>Electric Bills (2022-06-01)</b></p> <p>The value is higher than expected ):</p> <p>Show Results</p>
<p><b>Electric Bills (2023-01-01)</b></p> <p>The value is higher than expected ):</p> <p>Show Results</p>	<p><b>Phone Bills (2022-02-01)</b></p> <p>The value is higher than expected ):</p> <p>Show Results</p>
<p><b>Phone Bills (2022-05-01)</b></p> <p>The value is lower than expected (:</p> <p>Show Results</p>	<p><b>Phone Bills (2022-06-01)</b></p> <p>The value is higher than expected ):</p> <p>Show Results</p>
<p><b>Phone Bills (2022-08-01)</b></p> <p>The value is lower than expected (:</p>	<p><b>Phone Bills (2022-09-01)</b></p> <p>The value is lower than expected (:</p>

**Figure 4.16: Notification interface**

## **5. CHAPTER 5 TESTING AND EVALUATION**

### **5.1. INTRODUCTION**

The testing phase is the fourth phase in the waterfall model and a main phase in the Agile methodology, testing is the process of checking software's functions, and handling and correcting errors.

Testing and evaluation play a crucial role in various fields, including technology, manufacturing, software development, education, and more. This chapter provides an overview of testing and evaluation processes, their significance, and their applications in different domains. The chapter aims to familiarize readers with the fundamental concepts and methodologies involved in testing and evaluation, highlighting their importance in ensuring quality, reliability, and efficiency of products, systems, and processes.

### **5.2. TEST PLAN AND OBJECTIVES**

Our testing plan is divided into:

1. Module testing: is a white box type, where each component or subsystem will be tested separately and ensure that it is work probably, discover errors and correct the subsystem.
2. Integration testing: after finishing module testing, all system parts and subsystems need to be collected together to create the overall system and all modules communicate correctly without any conflicts.
3. Function testing: where all functional requirements must be satisfied correctly without errors or conflicts (all modules will be tested).

Objectives:

1. Understanding Testing: The chapter begins by defining testing and its purpose. It explains how testing is employed to assess the functionality, performance, and compliance of a product or system with specified requirements. Readers will gain insights into the objectives, benefits, and challenges associated with testing.

2. **Types of Testing:** This section explores various types of testing techniques and methodologies employed in different domains. It covers functional testing, usability testing, performance testing, security testing, and more. Each type is explained with relevant examples and use cases.
3. **Test Planning and Execution:** Effective test planning and execution are crucial for a successful testing process. This part of the chapter delves into the key steps involved in test planning, including test strategy development, test case creation, and test environment setup. It also discusses the execution of test cases, test data management, and the importance of test documentation.
4. **Test Automation and Tools:** With the increasing complexity of products and systems, test automation has become essential. This part of the chapter explores the benefits and challenges of test automation and highlights popular testing tools and frameworks available in the market.
5. **Testing in Agile and DevOps Environments:** Agile development methodologies and DevOps practices have transformed software development and testing approaches. This section discusses how testing fits into the Agile and DevOps frameworks, emphasizing the importance of continuous testing and integration throughout the development lifecycle.

### 5.3. STAGES OF TESTING

#### 5.3.1. WHITE BOX TESTING

##### 5.3.1.1. UNIT TESTING

Unit testing is a type of testing methodology that focuses on testing individual units of code to ensure their correctness and functionality in isolation. In software development, a unit refers to the smallest testable component of code, typically a function, method, or procedure. Unit testing involves writing test cases specifically designed to validate the behavior and output of these individual units [9].

**Register module units testing:**



**Table 5.1: Register module unit testing**

Unit	Inputs	Expected result	Testing result
Check-email	Email: faldossary@kku.edu.sa	False	Success
Check-email	Email: notregistered@kku.edu.sa	True	Success
Check-phone	Phone: 54123456	False	Success
Check-phone	Phone: 54123457	True	Success

**Login module units testing:****Table 5.2: Login module unit testing**

Unit	Inputs	Expected result	Testing result
Check-email	Email: faldossary@kku.edu.sa	1	Success
Check-email	Email: notregistered@kku.edu.sa	0	Success
Check-password	password: wrong	0	Success
Check-password	password: 123456	1	Success

**AI module units testing****Table 5.3: AI module units testing**

Unit	Inputs	Expected result	Testing result
Electricity bill prediction	Living: apartment Season: summer	500 ( $\pm 10\%$ )	Success
Electricity bill prediction	Living: apartment Season: spring	400 ( $\pm 10\%$ )	Success

Electricity prediction	bill	Living: apartment Season: autumn	300 ( $\pm 10\%$ )	Success
Electricity prediction	bill	Living: apartment Season: winter	200 ( $\pm 10\%$ )	Success
Electricity prediction	bill	Living: villa Season: summer	950 ( $\pm 10\%$ )	Success
Electricity prediction	bill	Living: villa Season: spring	850 ( $\pm 10\%$ )	Success
Electricity prediction	bill	Living: villa Season: autumn	750 ( $\pm 10\%$ )	Success
Electricity prediction	bill	Living: villa Season: winter	650 ( $\pm 10\%$ )	Success
Electricity prediction	bill	Living: commercial Season: summer	1500 ( $\pm 10\%$ )	Success
Electricity prediction	bill	Living: commercial Season: spring	1300 ( $\pm 10\%$ )	Success
Electricity prediction	bill	Living: commercial Season: autumn	1100 ( $\pm 10\%$ )	Success
Electricity prediction	bill	Living: commercial Season: winter	900 ( $\pm 10\%$ )	Success
Phone bill prediction		Internet: DSL	120 ( $\pm 10\%$ )	Success
Phone bill prediction		Internet: 5G/4G	150 ( $\pm 10\%$ )	Success
Phone bill prediction		Internet: Fiber	250 ( $\pm 10\%$ )	Success
Gas bill prediction		Duration: month	50 ( $\pm 10\%$ )	Success
Gas bill prediction		Duration: year	600 ( $\pm 10\%$ )	Success
Water bill prediction		Duration: month	150 ( $\pm 10\%$ )	Success

Water bill prediction	Duration: year	1600 ( $\pm 10\%$ )	Success
-----------------------	----------------	---------------------	---------

### 5.3.1.2. INTEGRATION TESTING

Integration testing is a software testing technique that focuses on verifying the interactions and interfaces between different components or modules of a system. It aims to identify defects that may arise when integrating these components and ensure that they work together correctly as a cohesive unit.

Integration testing is performed after unit testing and before system testing. While unit testing focuses on testing individual units of code in isolation, integration testing examines the behavior and functionality of interconnected units when combined. It helps identify issues such as communication failures, data inconsistencies, interface mismatches, and integration errors that may occur when components interact with each other [10].

#### User authentication and dashboard integration

**Table 5.4: User authentication and dashboard integration**

Test condition	Action	Expected result	Action result
Login (success scenario)	User submit login form with correct credential	The main dashboard page is opened	Successful Result
Login (un-success scenario)	User submit login form with un-correct credential	The main dashboard page is not opened and an error message appears	Successful Result
Login (success scenario)	User submit login form with correct credential	Open user's statistics	Successful Result
Login (un-success scenario)	User submit login form with un-correct credential	Could not open user' statistics	Successful Result

## Add bill and AI modules integration

**Table 5.5: Add bill and AI modules integration**

Test condition	Action	Expected result	Action result
Add bill (success scenario)	User submit add bill form with complete information	The bill is created and the browser is redirect to AI corresponding module	Successful Result
Add bill (un-success scenario)	User submit add bill form with missing information	No bill is created and no redirect to the corresponding AI module, an error message appears	Successful Result

## AI modules and result modules integration

**Table 5.6: AI modules and result modules integration**

Test condition	Action	Expected result	Action result
Bill cost prediction (success scenario)	The browser was directed to the AI module after adding bill	A notification about the result of submitted bill	Successful Result

### 5.3.2. BLACK BOX TESTING

#### 5.3.2.1. SYSTEM TESTING

System testing is a level of software testing that focuses on evaluating the entire system as a whole. It involves testing the integrated system to verify that it meets the specified requirements and functions correctly in its intended environment. System testing is performed after integration testing and prior to user acceptance testing.

The primary objective of system testing is to ensure that the software system behaves as expected and meets the functional, performance, security, and reliability requirements defined for

it. It tests the system as a black box, without detailed knowledge of its internal workings, to validate its behavior and performance against the desired outcomes.

**Table 5.7: System testing results**

<b>Functional requirements</b>	<b>Testing result</b>
Login	Successful Result
Logout	Successful Result
Registration	Successful Result
Add / remove/ view bills	Successful Result
Add/ Update / view budget	Successful Result
Add / update / view saving target	Successful Result
Receive add bill result	Successful Result
Generate statistics	Successful Result
View reports	Successful Result
Receive notifications	Successful Result
View previous notification	Successful Result

### **5.3.2.2. ACCEPTANCE TESTING**

Acceptance testing is a software testing technique performed to determine whether a system meets the requirements and criteria specified by stakeholders and end-users. It focuses on validating the system's functionality, usability, performance, and compliance with business needs, ensuring that it is ready for deployment and use in the production environment.

Acceptance testing is typically the final phase of the testing process, conducted after system testing. It involves stakeholders, end-users, or representatives from the intended user community who actively participate in testing and provide feedback on the system's suitability for acceptance.

**Table 5.8: Acceptance testing result**

<b>Non-Functional requirements</b>	<b>Testing result</b>
Ease of use: The design should be elegant, clear and uncomplicated.	Accepted

Reliability: The information in the site must be reliable and accurate.	Accepted
Availability: The website must be available at any time and serve the largest number of users at the same time.	Accepted
Maintainability: to allow the development of website, the addition of new features, and the development of services.	Accepted
Security: The site must be protected from unauthorized access to the system and its stored data. The password is stored as hashing not plain text	Accepted

## 5.4. TEST CASES AND TEST RESULTS

### 5.4.1. USER LOGIN

**Table 5.9: User login test case**

Test condition	Action	Expected result	Action result
Login with the correct username and password	user open login page, user logout, then submit login form	Login success and access the main website page	Successful Result
Login with an incorrect username or password	user open login page, user logout, then submit login form	Login fail	Successful Result

### 5.4.2. ADD BILL

**Table 5.10: Add bill test case**

Test condition	Action	Expected result	Action result
Accept request from reference PHP script add-bill-(bill_type).php	The Python file is requested from the PHP form	Connection success and read the request parameter bill_id	Successful Result

Load electric linear regression	Load the stored electric.pkl file	The stored electric linear regression will be loaded successfully without errors	Successful Result
Load phone linear regression	Load the stored phone.pkl file	The stored phone linear regression will be loaded successfully without errors	Successful Result
Load water linear regression	Load the stored water.pkl file	The stored water linear regression will be loaded successfully without errors	Successful Result
Load gas linear regression	Load the stored gas.pkl file	The stored gas linear regression will be loaded successfully without errors	Successful Result
Redirect to result-(bill_type).php	After complete prediction of bill cost redirect to result-(bill_type).php	Redirect to result-(bill_type).php with the correct result response value	Successful Result
Receive notification	Loading result-(bill_type).php	View correct notification about the status of the added bill	Successful Result

### 5.4.3. LINEAR REGRESSION RESULTS

We have four linear regression models, one for each type of bills:

1. Electric linear regression accuracy: 93.65 %
2. Phone linear regression accuracy: 90.61 %
3. Water linear regression accuracy: 97.50 %
4. Gas linear regression accuracy: 97.67 %

## 5.5. SUMMARY

The testing chapter provides an overview of different testing methodologies and techniques employed in software development. It covers the essential concepts of testing, including its purpose, benefits, and various levels of testing.

Unit testing, the first level of testing, focuses on testing individual units of code in isolation to ensure their correctness and functionality. It involves writing test cases specifically designed to validate the behavior and output of these units.

Integration testing comes next, aiming to verify the interactions and interfaces between different components or modules of a system. It helps identify defects that may arise when integrating these components and ensures they work together correctly as a cohesive unit.

System testing follows integration testing and focuses on evaluating the entire system as a whole. It validates the system's behavior, functionality, performance, and compliance with requirements, ensuring it meets the desired outcomes in its intended environment.

Acceptance testing, the final phase, involves stakeholders or end-users actively participating in testing to determine whether the system meets their needs and expectations. It focuses on validating the system's suitability for acceptance and deployment in the production environment.

Throughout the testing process, various test techniques, test environments, test cases, and tools are used to uncover defects, ensure quality, and provide feedback for improvements. Testing helps identify and address issues early in the development cycle, ensuring the reliability, functionality, and user satisfaction of the software system.



## **6. CHAPTER 6 CONCLUSION AND FUTURE WORK**

### **6.1. CONCLUSION**

In this project, we analysed, designed, implemented and tested web application to manage monthly and yearly bills in kingdom of Saudi Arabia, where we build AI model to predict the correct and accepted bill cost value according to our generated artificial datasets. The proposed system is a website implemented using web technologies, HTML5, CSS, JavaScript, Bootstrap, data table and jQuery for the front-end side and PHP with MySQL database, and Python for the server side.

We used full stack development server XAMPP which includes:

- Apache server
- PHP scripting programming language
- MySQL database
- CGI component to allow running external scripts like Python

We used linear regression to build four models, one model for each type of bills, electric, phone, water, and gas. The performance of our models is high and accurate.

The results showed that the proposed system satisfies all functional requirements, without errors, conflicts, or crashes.

### **6.2. LIMITATIONS OF THE PROJECT**

- Dataset is artificial and not real data, because we did not find real data in this field.
- The site is not online at this moment
- The website is not connected to the websites of the service providers

### **6.3. FUTURE ENHANCEMENTS**

- Collect real dataset about bills in kingdom of Saudi Arabia.
- Provide admin to manage and maintain the website's content, functionality, and security.
- Enhance AI by using deep learning.
- A bBills can offer the option to set up automatic payments to schedule the process.

## REFERENCES

- [1] <https://play.google.com/store/apps/details?id=sa.com.se.alkahraba&hl=ar&gl=US>
- [2] [https://play.google.com/store/apps/details?id=com.mint&hl=en\\_US&gl=US](https://play.google.com/store/apps/details?id=com.mint&hl=en_US&gl=US)
- [3] <https://apps.microsoft.com/store/detail/phone-bill-analyzer/9NBLGGH4Z0NQ?hl=en-us&gl=us&ocid=badge&rtc=1>
- [4] Gunasekaran, A. (1998). Agile manufacturing: enablers and an implementation framework. *International journal of production research*, 36(5), 1223-1247.
- [5] Pan, T., Zhao, J., Wu, W., & Yang, J. (2020). Learning imbalanced datasets based on SMOTE and Gaussian distribution. *Information Sciences*, 512, 1214-1233.
- [6] Aalen, O. O. (1989). A linear regression model for the analysis of life times. *Statistics in medicine*,
- [7] Dietterich, T. (1995). Overfitting and undercomputing in machine learning. *ACM computing surveys (CSUR)*, 27(3), 326-327.
- [8] Assaleh, K., & El-Hag, A. H. (2008, April). Estimating transformer oil parameters using polynomial networks. In *2008 International Conference on Condition Monitoring and Diagnosis* (pp. 1335-1338). IEEE.
- [9] Runeson, P. (2006). A survey of unit testing practices. *IEEE software*, 23(4), 22-29.
- [10] Hartmann, J., Imoberdorf, C., & Meisinger, M. (2000, August). UML-based integration testing. In *Proceedings of the 2000 ACM SIGSOFT international symposium on Software testing and analysis* (pp. 60-70).