VISVESVARAYA TECHNOLOGICAL UNIVERSITY
"Jnana Sangama," BELAGAVI-590 018.

A MINI PROJECT REPORT ON

"Electric Bill Generator"
Submitted by Partial Fulfillment of Requirements for the Course

of

"File Structures Laboratory with Mini Project[18ISL67]"
of Sixth Semester of Bachelor of Engineering in Information science & Engineering during the Academic Year 2022-23

Submitted by

Aman Arvind Kumar [4MH20IS012]   Manjunath K J [4MH20IS045]

&

Vijay N C [4MH20IS104]

Under the Guidance of

Prof. DHARMARAJ K B
Assistant Professor &
Department of ISE

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

MAHARAJA INSTITUTE OF TECHNOLOGY MYSORE
Belawadi S.R. Patna Taluk, Mandya Dist.- 571477
2022-2023

Maharaja Education Trust (R), Mysuru

Maharaja Institute of Technology Mysore

Belawadi, Sri Ranga Pattana Taluk, Mandya – 571 477
Approved by AICTE, New Delhi,
Affiliated to VTU, Belagavi & Recognized by Government of Karnataka

Department of Information Science and Engineering

<u>CERTIFICATE</u>

Certified that the mini project work entitled "**Electric Bill Generator**" has been successfully carried out by **Aman Arvind Kumar [4MH20IS012], Manjunath K J [4MH20IS045] and Vijay N C [4MH20IS104].** Bonafide students of **Maharaja Institute of Technology Mysore** in partial fulfillment of the requirements of **File Structures Laboratory with Mini Project[18ISL67] with Mini Project in Information Science and Engineering Department of Visvesvaraya Technological University, Belagavi** during the academic year 2022-2023. It is certified that all corrections/suggestion indicated for the Internal Assessment have been incorporated in the report deposited in the department library. The mini project report has been approved as it satisfies the academic requirements with respect to the mini project work prescribed for Bachelor of Engineering Degree.

Signature of Guide
Prof. Dharmaraj K B
Assistant Professor
Department of ISE, MITM

Signature of HOD
Dr. Sharath Kumar Y H
Professor & Head of Department
Department of ISE, MITM.

Name of the Examiners	Signature with date
1.	1.
2.	2.

ACKNOWLEDGEMENT

We sincerely owe out gratitude to all the persons who helped and gratitude us in completing this mini project work .

We are thankful to **Dr. B.G. Naresh Kumar, Principal, Maharaja Institute of Technology Mysore**, for having supported us in our academic Endeavor.

We are extremely thankful to **Dr. Sharath Kumar Y H, Professor, HOD Department of Information Science and Engineering**, for his support and his timely inquiries into the progress of the work.

We are greatly indebted to our guide **Prof. Dharmaraj K B** Assistant Professor in **Department of Information Science and Engineering**, for providing relevant information, valuable guidance, and encouragement to complete this report.

We are obliged to all teaching and non-teaching staff members of the Department of Information Science and Engineering**,** for the valuable information and support.

We are always thankful to our parents for their valuable support and guidance in every step.

We express our deepest gratitude and indebted thanks to MIT which has provided us an opportunity our most cherished desire of reaching our goal.

AMAN ARVIND KUMAR[4MH20IS012]

MANJUNATH K J [4MH20IS045]

VIJAY N C [4MH20IS104]

__ABSTRACT__

The Electric Bill Generator is an advanced software application that automates the process of generating accurate and comprehensive electric bills for users. It provides a user-friendly interface for managing and analyzing electricity consumption, empowering users to make informed decisions for efficient energy management. At its core, the Electric Bill Generator abstracts the complex calculations involved in billing and presents users with a simplified view of their energy consumption and associated costs. It encapsulates the intricate logic of tariff calculations, time-of-use rates, and discounts, shielding users from the complexities of billing computations. Through abstraction, the Electric Bill Generator offers a high-level representation of electricity consumption, transforming raw meter readings into meaningful information. It aggregates and organizes consumption data, allowing users to visualize their energy usage patterns, identify areas of high consumption, and evaluate the impact of energy-saving initiatives. The abstraction layer of the Electric Bill Generator conceals the underlying intricacies of data processing, database management, and system integration. It provides users with an intuitive interface that allows for easy input of consumption data, displays clear billing information, and presents insightful analytics in a user-friendly manner. By abstracting the billing process, the Electric Bill Generator simplifies and automates tasks that were previously manual and error-prone. It frees users from the burden of complex calculations, streamlines the billing process, and ensures accurate and transparent billing information.Additionally, the abstraction allows for customization and scalability. Users can configure billing parameters based on their specific needs, such as different tariff structures or time-of-use rates. The Electric Bill Generator is designed to handle varying user profiles, from residential customers to commercial and industrial entities, providing flexibility and scalability as energy management needs evolve.

INDEX:

CHAPTER: 01
<u>INTRODUCTION</u>

1.1 Overview of the Project:

An electric bill generator is a device or software application that simulates or calculates the amount of money a consumer or household would be billed for their electricity usage. It takes into account various factors such as the electricity consumption, applicable rates, and any additional charges or fees.

The purpose of an electric bill generator is to provide an estimate or representation of what the actual electricity bill would be for a given period. It can be used for budgeting purposes, helping individuals or businesses plan and manage their energy expenses.

Electric bill generators typically require input data such as the electricity consumption in kilowatt-hours (kWh) or the power consumption of specific appliances, the billing period, the applicable tariff structure (including time-of-use rates if applicable), and any fixed charges or taxes. Using this information, the generator applies the appropriate calculations and algorithms to determine the total amount due.

1.2 Problem Statement:

Designing an Electric Bill Generator for efficient management and analysis of electricity consumption.

The objective of this project is to develop an Electric Bill Generator that can generate accurate and comprehensive electric bills for users. The current manual process of generating electric bills is time-consuming, prone to errors, and lacks the ability to provide detailed consumption insights. This project aims to address these challenges by automating the billing process and introducing advanced features for effective energy management.

Key Challenges:
1. Automation: Develop a system that automates the process of generating electric bills, eliminating the need for manual calculations and reducing human errors.

2. Accuracy: Ensure that the generated bills accurately reflect the actual consumption by incorporating accurate meter readings and tariff rates.

3. Customization: Allow users to configure the billing parameters according to their specific requirements, such as different tariff structures, time-of-use rates, and discounts.

4. Energy Consumption Analysis: Implement features that enable users to analyze their energy consumption patterns, identify areas of high consumption, and make informed decisions for energy efficiency.

5. User-Friendly Interface: Design an intuitive and user-friendly interface that allows easy input of consumption data, displays clear billing information, and provides access to consumption analytics.

6. Scalability: Develop a system that can handle a large volume of consumption data and generate bills efficiently, ensuring it can scale up to meet the needs of various user profiles, including residential, commercial, and industrial customers.

7. Security: Implement robust security measures to protect sensitive customer data and ensure privacy during data transmission and storage.

By addressing these challenges, the Electric Bill Generator will streamline the billing process, improve accuracy, empower users with consumption insights, and ultimately contribute to more efficient energy management.

1.3 The Solution:

- Automated Billing Process: Develop an algorithm that automatically calculates electric bills based on meter readings and predefined tariff rates. Implement an automated data input system that retrieves meter readings from smart meters or allows users to manually enter readings.

- Tariff Configuration: Provide a user-friendly interface that allows users to configure billing parameters, including tariff rates, time-of-use rates, peak and off-peak hours, and any applicable discounts or surcharges. This customization feature ensures accurate billing based on the specific needs of each user.

- Consumption Analytics: Incorporate data analysis and visualization tools to enable users to analyze their energy consumption patterns. Generate reports and graphs that highlight peak usage periods, identify energy-intensive appliances, and suggest energy-saving measures.

- Billing Transparency: Ensure transparency in the billing process by clearly itemizing charges, including consumption charges, taxes, fixed fees, and any other applicable charges. This helps users understand their electricity costs and promotes trust in the billing system.

- Multiple User Profiles: Design the Electric Bill Generator to cater to different user profiles, such as residential, commercial, and industrial customers. Accommodate diverse billing structures, such as net metering for solar panel owners or demand-based billing for commercial entities.

- Scalability and Performance: Develop the system to handle a large volume of consumption data efficiently. Optimize algorithms and database structures to generate bills quickly and handle concurrent user requests without compromising performance.

- Data Security and Privacy: Implement robust security measures to protect user data. Encrypt sensitive information during transmission and storage, and ensure compliance with data protection regulations. Implement user access controls to ensure only authorized individuals can access billing information.

- Integration with Smart Grid: Integrate the Electric Bill Generator with smart grid technologies, such as advanced metering infrastructure (AMI) and demand response systems. This integration allows real-time data collection and enables dynamic pricing models based on grid conditions.

- Billing Reminders and Notifications: Implement a notification system that alerts users about upcoming bill due dates, payment reminders, and any changes in tariff rates or billing policies. This helps users stay informed and avoid late payments.

- Mobile and Web Compatibility: Develop mobile and web applications for the Electric Bill Generator, ensuring accessibility from various devices. This allows users to view and manage their bills conveniently from anywhere, enhancing user experience and convenience.

By implementing these solutions, the Electric Bill Generator will streamline the billing process, improve accuracy, provide valuable consumption insights, and empower users to make informed decisions for efficient energy management.

- Backup and Recovery:
    - Establish a regular backup schedule to ensure that files are backed up at defined intervals.
    - Test the backup and recovery process periodically to ensure its effectiveness.
    - Consider implementing version control mechanisms to track and restore previous versions of files if needed.

By implementing these solutions, you can address the common problems that may arise in a car rental system based on file structures, ensuring data integrity, performance, security, and reliability.

1.4 Existing System:

In this system user or customer will directly interact with the car owner and owner will decide whether the car is available or not. Then if it is available, he will give rent a car to the customer. The main drawback of this system is customer need to meet the car owner. this is time waste process.

1.5 Proposed System:

The proposal system in the Electric Bill Generator aims to provide users with personalized recommendations and suggestions to optimize their energy consumption and reduce electricity costs. It analyzes the user's historical consumption data and identifies areas where improvements can be made. Here's an outline of the proposal system:

1. Data Collection and Analysis:
    - Collect and store historical consumption data of users.
    - Analyze the data to identify consumption patterns, peak usage periods, and energy-intensive appliances.

2. Consumption Insights:
    - Present users with visualizations and reports that showcase their energy consumption trends.
    - Highlight areas of high consumption and identify potential energy-saving opportunities.

3. Energy Optimization Recommendations:
    - Based on the consumption analysis, provide personalized recommendations to users on how they

can optimize their energy usage and reduce costs.
- Offer suggestions such as adjusting thermostat settings, upgrading to energy-efficient appliances, or implementing smart home automation systems.

4. Cost Comparison and Forecasting:
   - Generate cost comparisons between different tariff structures, enabling users to evaluate potential savings by switching to a more suitable rate plan.
   - Provide forecasting tools to estimate future electricity costs based on consumption patterns and tariff changes.

5. Efficiency Tracking:
   - Implement a tracking system that allows users to monitor the impact of their energy-saving efforts over time.
   - Display energy efficiency metrics, such as savings achieved, reduction in carbon footprint, and payback period for energy-efficient investments.

6. Customized Energy Plans:
   - Enable users to create personalized energy plans based on their specific requirements and goals.
   - Provide options to set targets for energy reduction, cost savings, or environmental sustainability.

7. Education and Tips:
   - Offer educational resources, tips, and best practices to help users understand energy conservation techniques.
   - Provide guidance on energy-efficient behaviors, such as adjusting thermostat settings, utilizing natural light, and implementing power-saving practices.

8. Notifications and Reminders:
   - Send periodic notifications and reminders to users regarding energy-saving opportunities, tariff changes, or upcoming peak usage periods.
   - Alert users about potential energy waste or abnormalities in consumption patterns.

9. Integration with Smart Home Devices:
   - Integrate the Electric Bill Generator with smart home devices to provide real-time energy usage data and control features.
   - Allow users to monitor and control energy-consuming devices remotely, optimizing energy usage and reducing costs.

1.6 Advantages:

It is possible to store data compactly.

• Computer-based systems provide enhanced data retrieval techniques to retrieve data stored in files in easy and efficient way.

• It is easy to edit any information stored in computers in form of files.
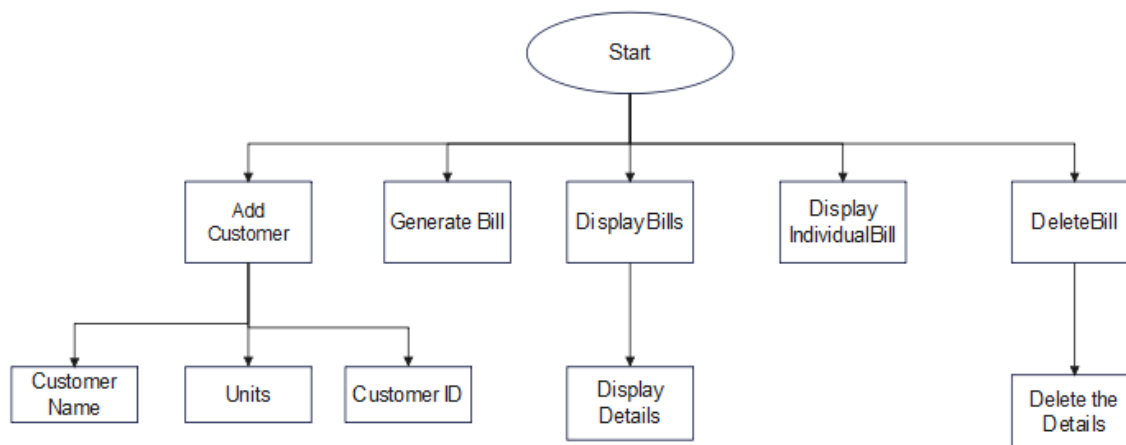
1.7 Data Flow Diagram:

Fig 1.7: Electric Bill Generator

CHAPTER:02

SOFTWARE REQUIREMENT SPECIFICATION

2.1 Software Requirements:

Hardware Requirements:

- Computer or Server ,Processor (CPU),Network Infrastructure.
- Scaling Considerations .
- Backup System.

Software Requirements:

- Programming Language: Choose a programming language that suits your project's needs. Popular options include Python, Java, C#, or PHP.
- Integrated Development Environment (IDE): Select an IDE that supports your chosen programming language. Examples include PyCharm, Eclipse, Visual Studio, or PhpStorm.

2.2.1 System Description:

C++ (Language used) C++ is a general-purpose programming language. It has imperative, object-oriented, and generic programming features, while also providing facilities for low level memory manipulation. It was designed with a bias towards system programming and embedded, resource constrained and large systems, with performance, efficiency, and flexibility of use as its design highlights.

C++ has also been found useful in many other contexts, with key strengths being software infrastructure and resource-constrained application, including desktop applications, servers (example: e-commerce, web search or SQL servers), and performance-critical applications (e.g.: telephone switches or space probes.).

C++ is a complied language, with implementations of it available on many platforms. Many vendors provide C++ compilers, including the free Software Foundation, Microsoft, Intel, and IBM.

CHAPTER: 03

FILE OPERATIONS

3.1 File Structure:

Operation	Input	Output
Insertion	Enter the Customer's Personal Details	Customer details are Saved Successfully to the file.
Display	Entering the choice which Car to be Select for a Rent.	Displaying the selected Car Details About their model, color, Maximum Speed, and Rent Price.
Searching	With the help of National ID of customer, we can search the details of customer which is present in the file.	If National ID is present in the file, it shows as Record found otherwise showing as Record not found.
Deleting	Deleting customer's Details.	Entered customers Details are Stored into Separate File in a System.

3.1.1 Primary Index:

Index on Sequential File, also called Primary Index, when the Index is associated to a Data File which is in turn sorted with respect to the search key.

- A Primary Index forces a sequential file organization on the Data File.
- Since a Data File can have just one order there can be just one Primary Index for Data File.

Usually used when the search key is also the primary key of the relation. Usually, these indexes fit in main memory.

Indexes on sequential files can be:

- Dense: One entry in the index file for every record in the data file.
- Sparse: One entry in the index file for each block of the data file.

3.2 Testing

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies and/or a finished product.

It is the process of exercising software with the intent of ensuring that a software system meets its requirements and user expectation does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement

3.2.1 Types of Testing

3.2.2 Unit Testing:

Unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage producers, and operating procedures, are tested to determine whether they are fit for use.

The primary goal of unit testing is to take the smallest piece of testable software in the application, isolate it from the reminder of the code, and determine whether it behaves exactly as you expect. Each unit is tested separately before integrating them into modules to test the interfaces between modules. Unit testing has proven its value in that a large percentage of defects are identified during its use.

3.2.3 Integrated Testing:

Integration testing is the logical extension of unit testing. In its simplest form, two units that have already been tested are combined into a component and the interface between them is tested. A component, in this sense, refers to an integrated aggregate of more than one unit. In a realistic scenario, many units are combined into components, which are in turn aggregated into even larger parts of the program.

The idea is to test combinations of pieces and eventually expand the process to test your modules with those of other groups. Eventually all the modules making up the process are tested together. Beyond that, if the program is composed of more than one process, they should be tested in pairs rather than all at once.

3.2.4 System Testing:

The process of performing a variety of tests on a system to explore functionality or to identify problems is called System Testing. It is usually required before and after a system is put in place. A series of systematic procedures are referred to while testing is being performed.

These procedures tell the tester how the system should perform and where common mistakes may be formed. Testers usually try to "Break the system" by entering data that may cause the system to malfunction or return incorrect information.

For example, A tester may put in a city in a search engine design to accept only states, to see how the system responds to the incorrect inputs.

3.2.5 Test Case:

Test case is a set of test inputs, execution and expected results developed for a particular objective. An excellent test case satisfies the following criteria:

- Reasonable probability of catching errors.
- Does interesting things.
- Does not do unnecessary things.
- Neither too simple nor too complex.
- Allows isolation and identification of errors.

3.2.6 Unit Testing:

Car Rental Module

Test Case	Test Case Name	Case Type	Input	Expected Results	Actual Results	Result
1	Insertion	Entering Customer's Details.	Entering Name, National Id.	Customer's Details are successfully saved.	Customer's Details are successfully saved.	Pass
2	Display	Details About Available cars.	When choice is entered.	Entered choice details should be displayed.	Entered choice details should be displayed.	Pass
3	Search	Validating Customer's.	When ID is entered.	Displays the searched ID.	Displays the searched ID.	Pass
4	Deleting	Deleting Customer's.	When detail's is entered.	Successfully delete to a File.	Successfully delete from File.	Pass

3.3 Modules

Add Customer:

```cpp
void addCustomer() {
    Customer customer;
    std::cout << "Enter customer name: ";
    std::cin.ignore();
    std::getline(std::cin, customer.name);
    std::cout << "Enter units consumed: ";
    std::cin >> customer.unitsConsumed;
    customer.billAmount = customer.unitsConsumed * 0.10; // Assuming the rate is $0.10 per unit

    // Generate unique ID
    std::string uniqueID = "C" + std::to_string(customers.size() + 1);
    customer.id = uniqueID;

    customers.push_back(customer);
    std::cout << "Customer added successfully! ID: " << uniqueID << std::endl;
}
```

Generate Bills:

```cpp
void generateBill() {
    std::string customerID;
    std::cout << "Enter customer ID: ";
    std::cin.ignore();
    std::getline(std::cin, customerID);

    auto it = std::find_if(customers.begin(), customers.end(), [&](const Customer& customer) {
        return customer.id == customerID;
    });

    if (it != customers.end()) {
        std::ofstream file("bills.txt", std::ios_base::trunc); // Open the file in truncation mode

        for (const auto& customer : customers) {
            file << "Customer ID: " << customer.id << std::endl;
            file << "Customer name: " << customer.name << std::endl;
            file << "Units consumed: " << customer.unitsConsumed << std::endl;
            file << "Bill amount: $" << customer.billAmount << std::endl;
            file << "-----------------------" << std::endl;
        }
        file.close();

        std::cout << "Bill generated and stored successfully!" << std::endl;
    } else {
        std::cout << "Customer not found!" << std::endl;
    }
}
```

Display Bills:

```cpp
void displayBills() {
    std::ifstream file("bills.txt");
    std::string line;
    while (std::getline(file, line)) {
        std::cout << line << std::endl;
    }
    file.close();

    if (file.eof()) {
        std::cout << "End of bill records." << std::endl;
    } else {
        std::cout << "Error occurred while reading bill records." << std::endl;
    }
}
```

Display Individual Bills:

```cpp
void displayIndividualBill() {
    std::string customerID;
    std::cout << "Enter customer ID: ";
    std::cin.ignore();
    std::getline(std::cin, customerID);

    std::ifstream file("bills.txt");
    std::string line;
    bool found = false;
    while (std::getline(file, line)) {
        if (line.find("Customer ID: " + customerID) != std::string::npos) {
            std::cout << line << std::endl;
            for (int i = 0; i < 3; i++) {
                std::getline(file, line);
                std::cout << line << std::endl;
            }
            found = true;
            break;
        }
    }
    file.close();

    if (!found) {
        std::cout << "Customer not found!" << std::endl;
    }
}
```

Delete Bill

```cpp
void deleteBill() {
    std::string customerID;
    std::cout << "Enter customer ID: ";
    std::cin.ignore();
    std::getline(std::cin, customerID);

    auto it = std::find_if(customers.begin(), customers.end(), [&](const Customer& customer) {
        return customer.id == customerID;
    });

    if (it != customers.end()) {
        customers.erase(it);
        std::cout << "Bill deleted successfully!" << std::endl;

        std::ofstream file("bills.txt");
        for (const auto& customer : customers) {
            file << "-----------------------" << std::endl;
            file << "Customer ID: " << customer.id << std::endl;
            file << "Customer name: " << customer.name << std::endl;
            file << "Units consumed: " << customer.unitsConsumed << std::endl;
            file << "Bill amount: " << customer.billAmount << std::endl;
            file << "-----------------------" << std::endl;
        }
        file.close();
    } else {
        std::cout << "Customer not found!" << std::endl;
    }
}
```

CHAPTER: 04

SNAPSHOTS

Add Customer :

Fig 4.1 Adding Customer Details

Generate Bill :

Fig 4.2 Generating Bill For Customer Details

.

Display Bills :

```
-----------------------
Electric Bill System
1. Add Customer
2. Generate Bill
3. Display Bills
4. Display Individual Bill
5. Delete Bill
6. Exit
-----------------------
Enter your choice: 3
Customer ID: C1
Customer name: kumar
Units consumed: 185
Bill amount: 1378.25
-----------------------
Customer ID: C2
Customer name: john
Units consumed: 210
Bill amount: 1564.5
-----------------------
End of bill records.
```
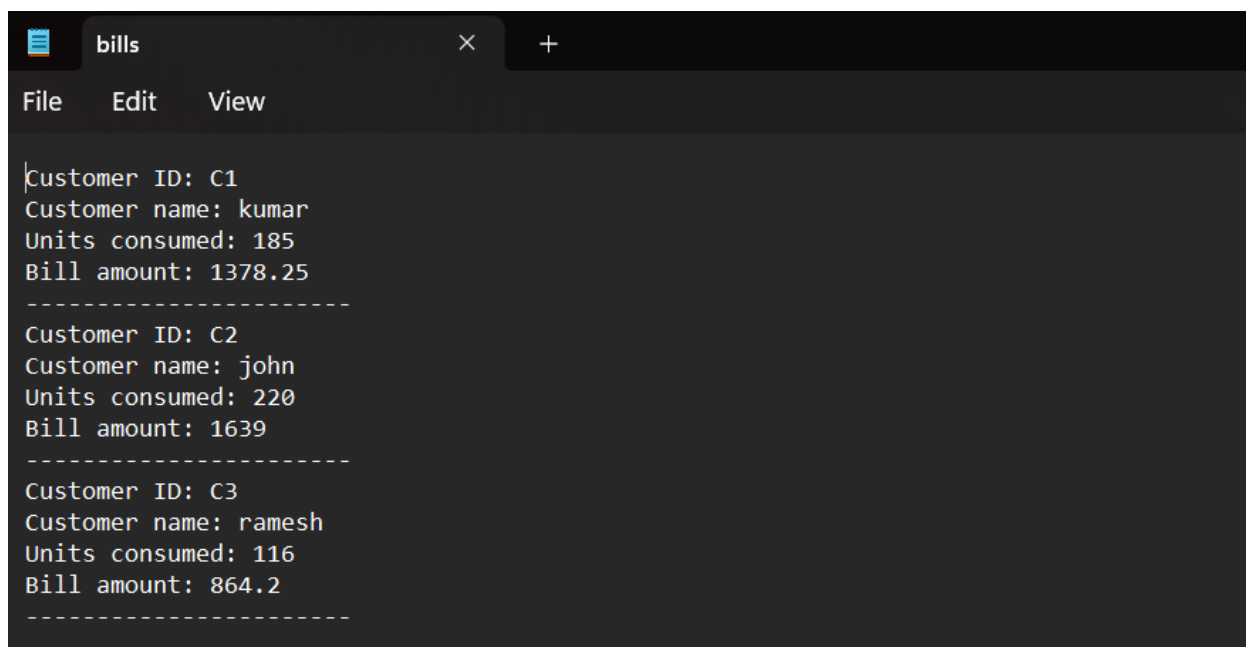
Fig:4.3 Showing the Bill of Every Customer.

Display Individual Bills :

```
-----------------------
Electric Bill System
1. Add Customer
2. Generate Bill
3. Display Bills
4. Display Individual Bill
5. Delete Bill
6. Exit
-----------------------
Enter your choice: 4
Enter customer ID: C1
Customer ID: C1
Customer name: kumar
Units consumed: 185
Bill amount: 1378.25
```

Fig:4.4 Showing the Individual Bill For Each Customer.

Delete Bill :

Fig:4.5 Deleting the Customer Based On the Customer ID.

Bills.txt File :

Fig:4.5 Bill Saved in the bills.txt File .

CONCLUSION AND FUTURE SCOPE

Car rental business has emerged with a new goody compared to the experience where every activity concerning car rental business is limited to a physical location only. Even though the physical location has not been totally eradicated, the nature of functions and how these functions are achieved has been reshaped by the power of internet. Now a days, customers can reserve cars online, rent car online, and have the car brought to their door step once the customer is a registered member or go to the office to pick the car.

The web-based car rental system has offered an advantage to both customers as well as Car Rental Company to manage the business and satisfies customer's need at the click of a button efficiently and effectively.

While making the system, an eye has been kept on making it as user-friendly, as cost-effective, and as flexible as possible. As such one may hope that the system will be acceptable to any user and will adequately meet his/her needs.

- It can be safely concluded that the product is a highly
- efficiently GUI base component.
- The system has reached a steady state where almost bugshave been eliminated.
- This component can be easily plugged in many others systems.

FUTURE SCOPE:

The car rental system presented in the project can be further improved and expanded with several future enhancements. Here are some potential areas for future development:

- **Online Reservation and Booking:** Implement an online platform where customers can browse available cars, check their availability in real-time, and make reservations and bookings directly through the system. This can include features like online payment integration, automated confirmation emails, and the ability to manage reservations and cancellations.

- **Integration with Mobile Apps:** Develop mobile applications for iOS and Android platforms, allowing customers to access the car rental system on their smartphones or tablets. Mobile apps can provide a more convenient and user-friendly experience, including features like push notifications, GPS-based car location, and seamless integration with device functionalities like contact lists and calendars.

- **Fleet Management:** Enhance the system to include fleet management capabilities for car rental companies. This can involve features such as tracking vehicle maintenance schedules, monitoring fuel consumption, managing vehicle inspections, and generating reports for better fleet management and optimization.

- **Loyalty Programs and Customer Profiles:** Implement a customer loyalty program that rewards frequent renters with discounts, exclusive offers, or loyalty points. Create customer profiles to store rental history, preferences, and personal details, allowing for personalized recommendations and improved customer service.

REFERENCES

TEXTBOOKS

The following books were very helpful during the completion of the project

- K.R. Venugopal, K.G. Srinivas, P.M. Krishna Raj: **File Structures Using C**++, Tata McGraw-Hill, 2008.
- Scot Robert Ladd: **C**++ **Components and Algorithms**, BPB Publications, 1993.
- Raghu Ramakrishna and Johannes Gehrke: **Database Management Systems**, 3rd Edition, McGraw Hill, 2003.

WEBSITES

1. C++ Tutorial – https://www.w3schools.com/cpp/
2. Dev C++ Tutorial – https://cplusplus.com/doc/tutorial/introduction/devcpp/
3. File System Structure – https://www.tutorialspoint.com/file-system-structure