



PRACTICAL

LIBRARY MANAGEMENT SCENARIO.

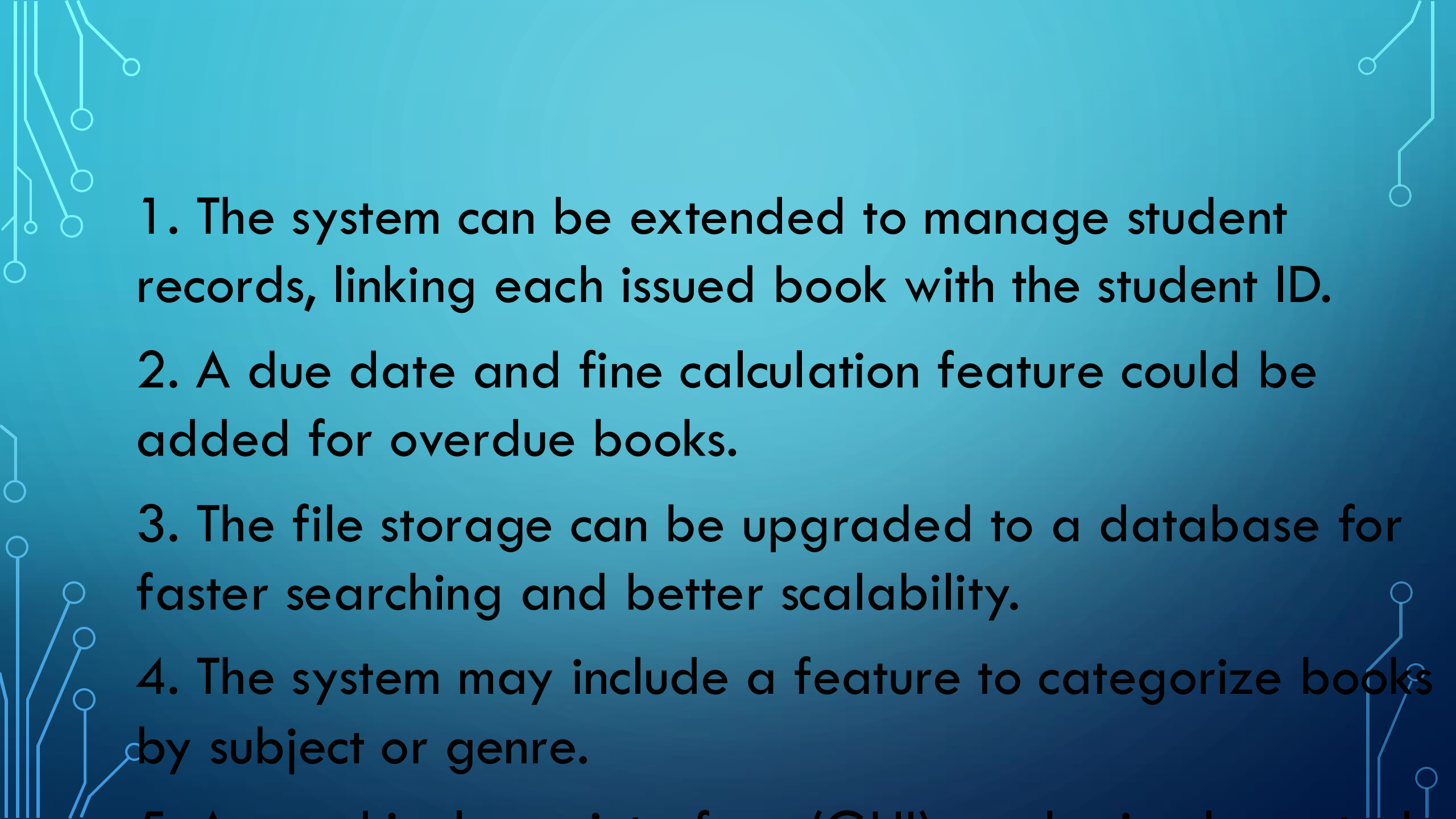
EMPLOYEE PAYROLL SYSTEM SCENARIO.

LIBRARY MANAGEMENT SYSTEM:

- ****Efficient***: The system should be able to quickly process book additions, issuances, and returns.*
- ****Organized***: The system should store book data in a structured and easily accessible manner.*
- ****Persistent***: The system should store book data in a file for long-term storage.*
- ****User-friendly***: The system should provide an intuitive interface for users to interact with.*
- ****Informative***: The system should display relevant statistics and information about the book collection.*

PROBLEM:

- 1. Library Management System
- Scenario:
 - Design a system to manage a library's book collection. The program should allow users to add new books, issue books to students, and track the return of borrowed books.
- Requirements:
 - Create a Book class with attributes like book ID, title, author, and availability status.
 - Implement methods to issue and return books.
 - Design a Library class to store a collection of books and provide a method to search for a book by title or ID.
 - Display statistics such as the total number of books, issued books, and available books.
 - Store the book data in a file for persistent storage.

- 
- The background of the slide is a solid blue color. It is decorated with white, stylized circuit board traces. These traces are located along the left and right edges, with some branching out towards the center. Small white circles, resembling solder points or vias, are placed at various intervals along these traces.
1. The system can be extended to manage student records, linking each issued book with the student ID.
 2. A due date and fine calculation feature could be added for overdue books.
 3. The file storage can be upgraded to a database for faster searching and better scalability.
 4. The system may include a feature to categorize books by subject or genre.

main.c... Output



```

1  #include <iostream>
2  #include <fstream>
3  #include <vector>
4  #include <string>
5  using namespace std;
6
7  // Book class
8  class Book {
9      int bookID;
10     string title;
11     string author;
12     bool available;
13
14 public:
15     Book() {}
16     Book(int id, string t, string a,
17         bool avail = true) {
18         bookID = id;
19         title = t;
20         author = a;
21         available = avail;
22     }
23     int getID() const { return bookID
24         ; }
25     string getTitle() const { return

```

main.c...

Output



```

24     string getTitle() const { return
25         title; }
26     string getAuthor() const { return
27         author; }
28     bool isAvailable() const { return
29         available; }
30
31     void issueBook() { available =
32         false; }
33     void returnBook() { available =
34         true; }
35
36     void display() const {
37         cout << "ID: " << bookID
38         << " | Title: " << title
39         << " | Author: " <<
40         author
41         << " | Status: " <<
42         (available ?
43         "Available" :
44         "Issued") << endl;
45     }
46
47     // Save book info to file
48     void saveToFile(ofstream &out)

```

main.c...

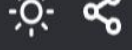
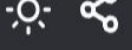
Output



```

39     void saveToFile(ofstream &out)
40         const {
41         out << bookID << "," << title
42         << "," << author << ","
43         << available << "\n";
44     }
45
46     // Load book info from file
47     static Book loadFromString(const
48         string &line) {
49         int id;
50         string t, a;
51         bool avail;
52         size_t pos1 = line.find(',');
53         size_t pos2 = line.find(',',
54             pos1 + 1);
55         size_t pos3 = line.find(',',
56             pos2 + 1);
57         id = stoi(line.substr(0, pos1
58             ));
59         t = line.substr(pos1 + 1,
60             pos2 - pos1 - 1);
61         a = line.substr(pos2 + 1,
62             pos3 - pos2 - 1);
63         avail = stoi(line.substr(pos3

```

```
55     avail = stor[line.substr(poss
        + 1));
```

```
56
57     return Book(id, t, a, avail);
58 }
59 };
```

```
60
61 // Library class
62 class Library {
63     vector<Book> books;
64
65 public:
66     void addBook(const Book &b) {
67         books.push_back(b);
68         cout << "Book added
        successfully!\n";
69     }
70
71     void issueBook(int id) {
72         for (auto &b : books) {
73             if (b.getID() == id) {
74                 if (b.isAvailable())
75                     b.issueBook();
76                 cout << "Book
        issued
        successfully!\n";
77     } else {
78         cout << "Book
        already issued!\n";
79     }
80     return;
81     }
82 }
83 cout << "Book not found!\n";
84 }
85
86 void returnBook(int id) {
87     for (auto &b : books) {
88         if (b.getID() == id) {
89             if (!b.isAvailable())
90                 b.returnBook();
91                 cout << "Book
        returned
        successfully!\n";
92     } else {
93         cout << "Book was
        not issued!\n";
94     }
95     return;
96 }
97 }
98 cout << "Book not found!\n";
99 }
100
101 void searchBookByID(int id) {
102     for (const auto &b : books) {
103         if (b.getID() == id) {
104             b.display();
105             return;
106         }
107     }
108     cout << "Book not found!\n";
109 }
110
111 void searchBookByTitle(string
    title) {
112     for (const auto &b : books) {
113         if (b.getTitle() == title
            ) {
114             b.display();
115             return;
116         }
117     }
118 }
```

```
75         b.issueBook();
76         cout << "Book
        issued
        successfully!\n";
77     } else {
78         cout << "Book
        already issued!\n";
79     }
80     return;
81     }
82 }
83 cout << "Book not found!\n";
84 }
85
86 void returnBook(int id) {
87     for (auto &b : books) {
88         if (b.getID() == id) {
89             if (!b.isAvailable())
90                 b.returnBook();
91                 cout << "Book
        returned
        successfully!\n";
92     } else {
93         cout << "Book was
        not issued!\n";
94     }
95     return;
96 }
97 }
98 cout << "Book not found!\n";
99 }
100
101 void searchBookByID(int id) {
102     for (const auto &b : books) {
103         if (b.getID() == id) {
104             b.display();
105             return;
106         }
107     }
108     cout << "Book not found!\n";
109 }
110
111 void searchBookByTitle(string
    title) {
112     for (const auto &b : books) {
113         if (b.getTitle() == title
            ) {
114             b.display();
115             return;
116         }
117     }
118 }
```

```

117     }
118     cout << "Book not found!\n";
119 }
120
121 void displayStats() {
122     int total = books.size();
123     int issued = 0, available = 0
        ;
124     for (const auto &b : books) {
125         if (b.isAvailable())
126             available++;
127         else
128             issued++;
129     }
130     cout << "Total Books: " <<
        total
131         << " | Available: " <<
            available
132         << " | Issued: " <<
            issued << endl;
133 }
134
135 void saveToFile(const string
    &filename) {
136     ofstream out(filename);

```

```

138         b.saveToFile(out);
139     }
140     out.close();
141 }
142
143 void loadFromFile(const string
    &filename) {
144     ifstream in(filename);
145     string line;
146     while (getline(in, line)) {
147         books.push_back(Book
            ::loadFromString(line
                ));
148     }
149     in.close();
150 }
151
152 void displayAllBooks() {
153     for (const auto &b : books) {
154         b.display();
155     }
156 }
157 };
158
159 // Main program

```




```

160 int main() {
161     Library lib;
162     lib.loadFromFile("books.txt");
163
164     int choice;
165     do {
166         cout << "\n--- Library
            Management System ---\n";
167         cout << "1. Add Book\n2.
            Issue Book\n3. Return
            Book\n4. Search by ID\n5.
            Search by Title\n";
168         cout << "6. Display All
            Books\n7. Display
            Stats\n8. Exit\n";
169         cout << "Enter choice: ";
170         cin >> choice;
171
172         if (choice == 1) {
173             int id;
174             string title, author;
175             cout << "Enter Book ID: ";
176             cin >> id;
177             cin.ignore();
178             cout << "Enter Title: ";
179             getline(cin, title);

```


main.c...




Output



```
178     cout << "Enter Title: ";
179     getline(cin, title);
180     cout << "Enter Author: ";
181     getline(cin, author);
182     lib.addBook(Book(id,
        title, author));
183 } else if (choice == 2) {
184     int id;
185     cout << "Enter Book ID to
        issue: ";
186     cin >> id;
187     lib.issueBook(id);
188 } else if (choice == 3) {
189     int id;
190     cout << "Enter Book ID to
        return: ";
191     cin >> id;
192     lib.returnBook(id);
193 } else if (choice == 4) {
194     int id;
195     cout << "Enter Book ID to
        search: ";
196     cin >> id;
197     lib.searchBookByID(id);
198 } else if (choice == 5) {
199     string title;
200     cin.ignore();
201     cout << "Enter Title to
        search: ";
202     getline(cin, title);
203     lib.searchBookByTitle
        (title);
204 } else if (choice == 6) {
205     lib.displayAllBooks();
206 } else if (choice == 7) {
207     lib.displayStats();
208 } else if (choice == 8) {
209     lib.saveToFile("books
        .txt");
210     cout << "Exiting... Data
        saved!\n";
211 } else {
212     cout << "Invalid
        choice!\n";
213 }
214 } while (choice != 8);
215
```

main.c...

Output






```
--- Library Management System ---
1. Add Book
2. Issue Book
3. Return Book
4. Search by ID
5. Search by Title
6. Display All Books
7. Display Stats
8. Exit
Enter choice: 1
Enter Book ID: k
Enter Title: Enter Author: Book added
successfully!

--- Library Management System ---
1. Add Book
2. Issue Book
3. Return Book
4. Search by ID
5. Search by Title
6. Display All Books
7. Display Stats
8. Exit
Enter choice: Enter Book ID: Enter Title:
Enter Author: Book added successfully!
```

main.c...

Output





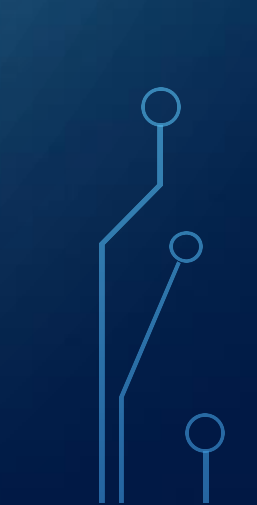


• 2. Employee Payroll System

• Scenario:

- Develop a simple payroll system for a company. The system should calculate and display the salary of employees based on their working hours and hourly rate.

• Requirements:

- Create a class Employee with attributes like name, ID, hours worked, and hourly rate.
 - Implement methods to calculate the total salary and generate a salary slip.
 - Provide functionality to input, update, and delete employee records.
 - Include a search feature to find employees by their ID.
 - Allow users to view a list of employees with their salaries and generate a summary report showing the total payroll amount.
- 
- 
- 

Employee Payroll System:

- ***Accurate***: The system should calculate salaries correctly based on working hours and hourly rates.
- ***Efficient***: The system should quickly process employee data and generate salary slips.
- ***User-friendly***: The system should provide an intuitive interface for users to input, update, and delete employee records.
- ***Informative***: The system should display detailed salary slips and provide a summary report of the total payroll amount.
- ***Flexible***: The system should allow users to easily search, update, and delete employee records.
- ***Secure***: The system should protect employee data from unauthorized access or modifications.
- ***Scalable***: The system should be able to handle a growing number of employees and payroll calculations.

main.c...

Output



main.c...

Output



main.c...

Output



```
1  #include <iostream>
2  #include <vector>
3  #include <string>
4  #include <iomanip>
5  using namespace std;
6
7  class Employee {
8  private:
9      string name;
10     int id;
11     double hoursWorked;
12     double hourlyRate;
13
14 public:
15     // Constructor
16     Employee(string n, int i, double
        hours, double rate)
17     : name(n), id(i), hoursWorked
        (hours), hourlyRate(rate)
        {}
18
19     // Getters
20     int getId() const { return id; }
21     string getName() const { return
        name; }
22     double getHoursWorked() const {
```

```
22     double getHoursWorked() const {
        return hoursWorked; }
23     double getHourlyRate() const {
        return hourlyRate; }
24
25     // Setters (for updating)
26     void setName(string n) { name = n
        ; }
27     void setHoursWorked(double h) {
        hoursWorked = h; }
28     void setHourlyRate(double r) {
        hourlyRate = r; }
29
30     // Calculate salary
31     double calculateSalary() const {
32         return hoursWorked *
        hourlyRate;
33     }
34
35     // Generate salary slip
36     void generateSalarySlip() const {
37         cout << "\n==== Salary Slip
        =====< endl;
38         cout << "Employee Name : " <<
        name << endl;
39         cout << "Employee ID   : " <<
```

```
39         cout << "Employee Name   : " <<
        name << endl;
40         cout << "Employee ID     : " <<
        id << endl;
41         cout << "Hours Worked    : " <<
        hoursWorked << endl;
42         cout << "Hourly Rate     : " <<
        hourlyRate << endl;
43         cout << "Total Salary   : " <<
        fixed << setprecision(2)
        << calculateSalary() <<
        endl;
44         cout << "
        =====\n
        " << endl;
45     }
46
47     class PayrollSystem {
48     private:
49         vector<Employee> employees;
50
51     public:
52         // Add employee
53         void addEmployee(const Employee&
        emp) {
54             employees.push back(emp);
```


main.c...		Output		main.c...		Output		main.c...		Output	
54	employees.push_back(emp);	72	emp.setName(newName);	90	}						
55	cout << "Employee added successfully!\n";	73	emp.setHoursWorked(newHours);	91	}						
56	}	74	emp.setHourlyRate(newRate);	92	cout << "Employee with ID " << id << " not found.\n";						
57				93	}						
58	// Update employee by ID	75		94							
59	void updateEmployee(int id) {	76	cout << "Employee updated successfully!\n";	95	// Search employee by ID						
60	for (auto &emp : employees) {		return;	96	void searchEmployee(int id) const {						
61	if (emp.getId() == id) {	77			{						
62	string newName;	78	}	97	for (const auto &emp : employees) {						
63	double newHours, newRate;	79			if (emp.getId() == id) {						
64	cout << "Enter new name: ";	80	cout << "Employee with ID " << id << " not found.\n";	98	cout << "\nEmployee Found:\n";						
65	cin.ignore();	81	}	99	emp						
66	getline(cin, newName);	82		100	.generateSalarySlip();						
		83	// Delete employee by ID		return;						
67	cout << "Enter new hours worked: ";	84	void deleteEmployee(int id) {	101							
		85	for (auto it = employees.begin(); it != employees.end(); ++it) {	102	}						
68	cin >> newHours;		if (it->getId() == id) {	103							
69	cout << "Enter new hourly rate: ";	86	employees.erase(it);	104	cout << "Employee with ID " << id << " not found.\n";						
70	cin >> newRate;	87	cout << "Employee deleted successfully!\n";	105	}						
71		88		106							
72	emp.setName(newName);		return;	107	// View all employees						
73	emp.setHoursWorked	89		108	void viewAllEmployees() const {						

```
109 cout << "\n=====\nEmployee\nList\n=====\n";
110 cout << left << setw(10) << "ID"
111 << setw(20) << "Name"
112 << setw(15) << "Hours\nWorked"
113 << setw(15) << "Hourly\nRate"
114 << setw(15) << "Salary"
115 << endl;
116 cout << "\n-----\n";
117 for (const auto &emp : employees) {
118     cout << left << setw(10) << emp.getId()
119 << setw(20) << emp.getName()
120 << setw(15) << emp.getHoursWorked()
121 << setw(15) << emp.getHourlyRate()
122 << setw(15) << emp.getSalary() << endl;
```

```
main.c... Output
122 << setw(15) << fixed << setprecision(2) << emp
123 << endl;
124 }
125 }
126 // Generate summary report
127 void generateSummaryReport()
128 {
129     const {
130     double totalPayroll = 0;
131     for (const auto &emp : employees) {
132         totalPayroll += emp.calculateSalary();
133     }
134     cout << "\n=====\nPayroll\nSummary\n=====\n";
135     cout << "Total Employees : " << employees.size() << endl;
136     cout << "Total Payroll : " << fixed << setprecision(2) << totalPayroll << endl;
```

```
main.c... Output
136 cout << "\n-----\n";
137 }
138 };
139 int main() {
140     PayrollSystem ps;
141     int choice;
142     do {
143         cout << "\n--- Employee\nPayroll System ---\n";
144         cout << "1. Add Employee\n";
145         cout << "2. Update\nEmployee\n";
146         cout << "3. Delete\nEmployee\n";
147         cout << "4. Search\nEmployee\n";
148         cout << "5. View All\nEmployees\n";
149         cout << "6. Generate Payroll\nSummary\n";
150         cout << "7. Exit\n";
151     } while (choice != 7);
152 }
```


main.c...

Output



main.c...

Output



main.c...

Output



```
152 cout << "7. Exit\n";
153 cout << "Enter your choice: "
    ;
154 cin >> choice;
155
156 switch (choice) {
157     case 1: {
158         string name;
159         int id;
160         double hours, rate;
161         cout << "Enter
            employee ID: ";
162         cin >> id;
163         cin.ignore(); //
            clear buffer
164         cout << "Enter name:
            ";
165         getline(cin, name);
166         cout << "Enter hours
            worked: ";
167         cin >> hours;
168         cout << "Enter hourly
            rate: ";
169         cin >> rate;
170         ps.addEmployee
            (Employee(name, id,
```

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

```
(Employee(name, id,
hours, rate));
```

break;

}

case 2: {

int id;

```
cout << "Enter
employee ID to
update: ";
```

cin >> id;

```
ps.updateEmployee(id
);
```

break;

}

case 3: {

int id;

```
cout << "Enter
employee ID to
delete: ";
```

cin >> id;

```
ps.deleteEmployee(id
);
```

break;

}

case 4: {

int id;

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

case 4: {

int id;

```
cout << "Enter
employee ID to
search: ";
```

cin >> id;

```
ps.searchEmployee(id
);
```

break;

}

case 5:

```
ps.viewAllEmployees
();
```

break;

case 6:

```
ps
.generateSummaryRep
ort();
```

break;

case 7:

```
cout << "Exiting
program...\n";
```

break;

default:

```
cout << "Invalid
choice! Please try
again.\n";
```




```
196         break;
197     case 6:
198         ps
            .generateSummaryRep
            ort();
199         break;
200     case 7:
201         cout << "Exiting
            program...\n";
202         break;
203     default:
204         cout << "Invalid
            choice! Please try
            again.\n";
205     }
206     } while (choice != 7);
207
208     return 0;
209 }
```

Run

--- Employee Payroll System ---

1. Add Employee
2. Update Employee
3. Delete Employee
4. Search Employee
5. View All Employees
6. Generate Payroll Summary
7. Exit

Enter your choice: 2

Enter employee ID to update: k

Employee with ID 0 not found.

--- Employee Payroll System ---

1. Add Employee
2. Update Employee
3. Delete Employee
4. Search Employee
5. View All Employees
6. Generate Payroll Summary
7. Exit

Enter your choice: Enter employee ID to

SUBMITTED BY
T.KARTHIKA
II BSC CS

