Topic: Online Laundry Service

Group no: MLB_WD_CSNE_13_09

Campus: Malabe

Submission Date: 2022/05/20

We declare that this is our own work and this assignment does not incorporate without acknowledgment any material previously submitted by anyone else in SLIIT or any other university/Institute. And we declare that each one of us equally contributed to the completion of this Assignment.

Registration No	**Name**	**Contact Number**
IT21258176	Sanuja Methmal	+94761936975
IT21337512	Thisun Senaratna	+94713556355
IT21266232	Induwara Ashinshana	+94764533491
IT21283376	Avishka Pramod	+94717301531
IT21245756	Pivithuru Milan	+94728925071

Terms of Reference

This report was prepared to fulfill the requirements of the module Object Oriented Concepts (OOC – IT1050) at the Sri Lanka Institute of Information Technology (SLIIT).

Acknowledgment

We would like to express our deepest appreciation to all those who provided us with the possibility to complete this report. Special gratitude must be given to our OOC Lecturer and the OOC lecturer-in-charge whose contribution helped to fulfill our project.

Table of Contents

Introduction

Modernization has changed the lifestyle of the society by a tremendous amount, in developing countries computerized laundry management systems are well accepted due to the elimination of flaws in traditional systems such as inefficiency of time, data inaccuracies, and errors in customer details. This report proposes an online laundry system which is more time efficient and convenient to the customer.

Requirement Analysis

- Create accounts for new customers.
- Verify login credentials of customers.
- Place orders online.
- Cancel already placed orders.
- Calculate the fee based on information provided.
- Track the order
- Assign employees for each task.
- Update status of each order.

Description of the requirements

The online laundry service allows customers to place online orders via the website of the laundry center whenever they require laundry service.

Prior to placing an order, new Customers must register by creating a user account while existing customers must login using their credentials.

The following details are required to be provided by the customer for placing an order.

- The service type. (laundry or dry cleaning must be chosen.)
- Approximate weight of the items.
- Pickup and delivery location.

The service charge is estimated based on the weight of the clothes and service type. The total charge will be automatically calculated by the system by adding taxes. That amount must be settled using an online payment processing method when placing the order. A staff member verifies the order details with the payment and confirms the order. From here onwards, customers can track the order status (confirmed, picked, processed, packed, shipped, or delivered) through the website. Customers can cancel the orders before it had been picked up by the laundry center.

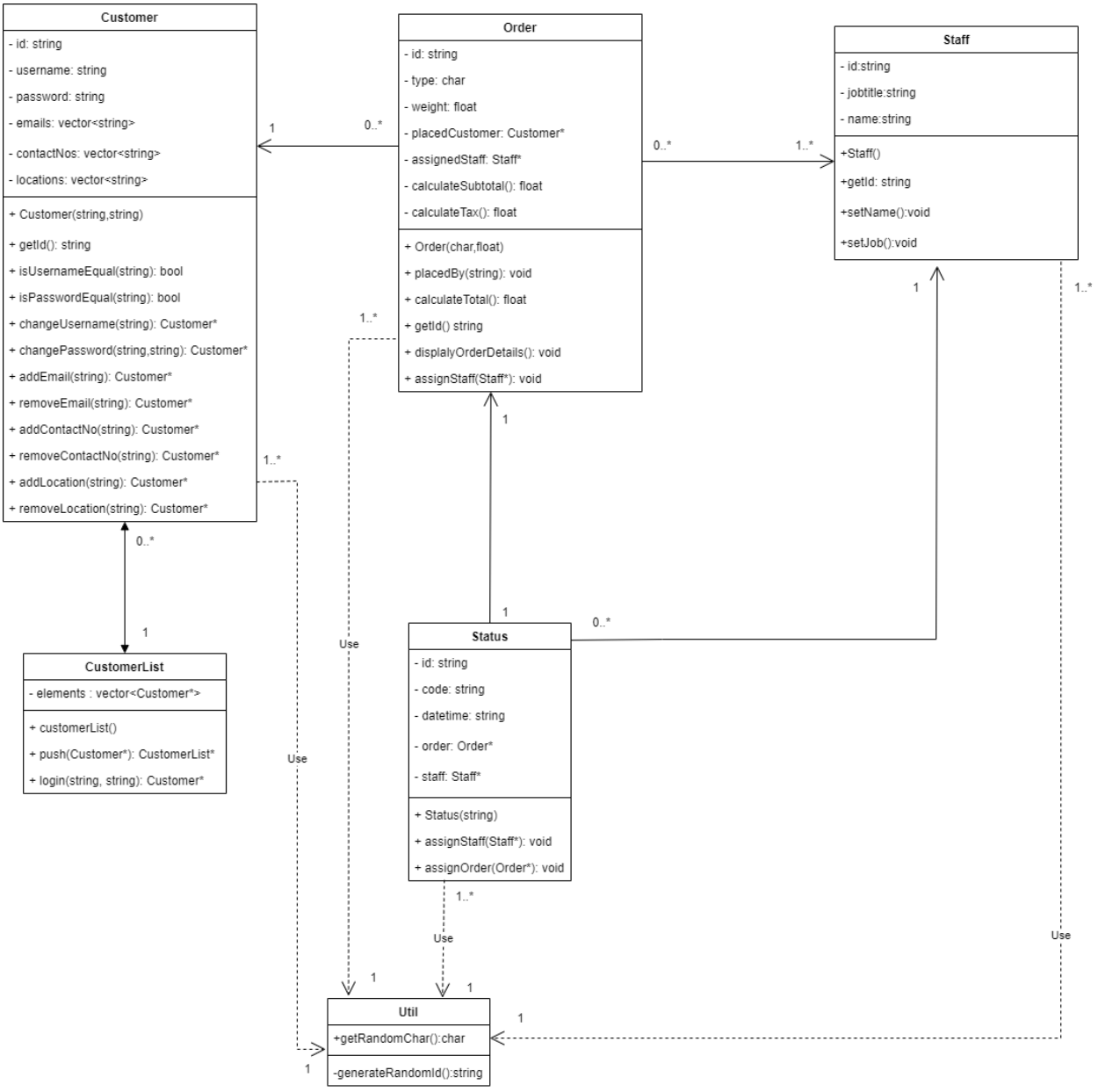
After order confirmation, the laundry sends an employee to inspect the items and bring them to the laundry center. Each order is assigned to multiple staff members who are responsible for updating the order status after completing the assigned task (pickup, cleaning, packing, delivery).

First, a staff member sanitizes the clothing items. Next, clothes are washed or dry cleaned based on the order type chosen by the customer. Then, clothes are folded and packaged prior to delivery. When the process inside the laundry is finished, the package is delivered back to the customer’s location.

Classes

- Customer
- CustomerList
- Order
- Staff
- Status
- Util

Class Diagram



CRC cards

Customer	
Responsibility	**Collaborators**
Store customer contact details and location. Store customer login credentials	

CustomerList	
Responsibility	**Collaborators**
Search user credentials for login purposes	Customer

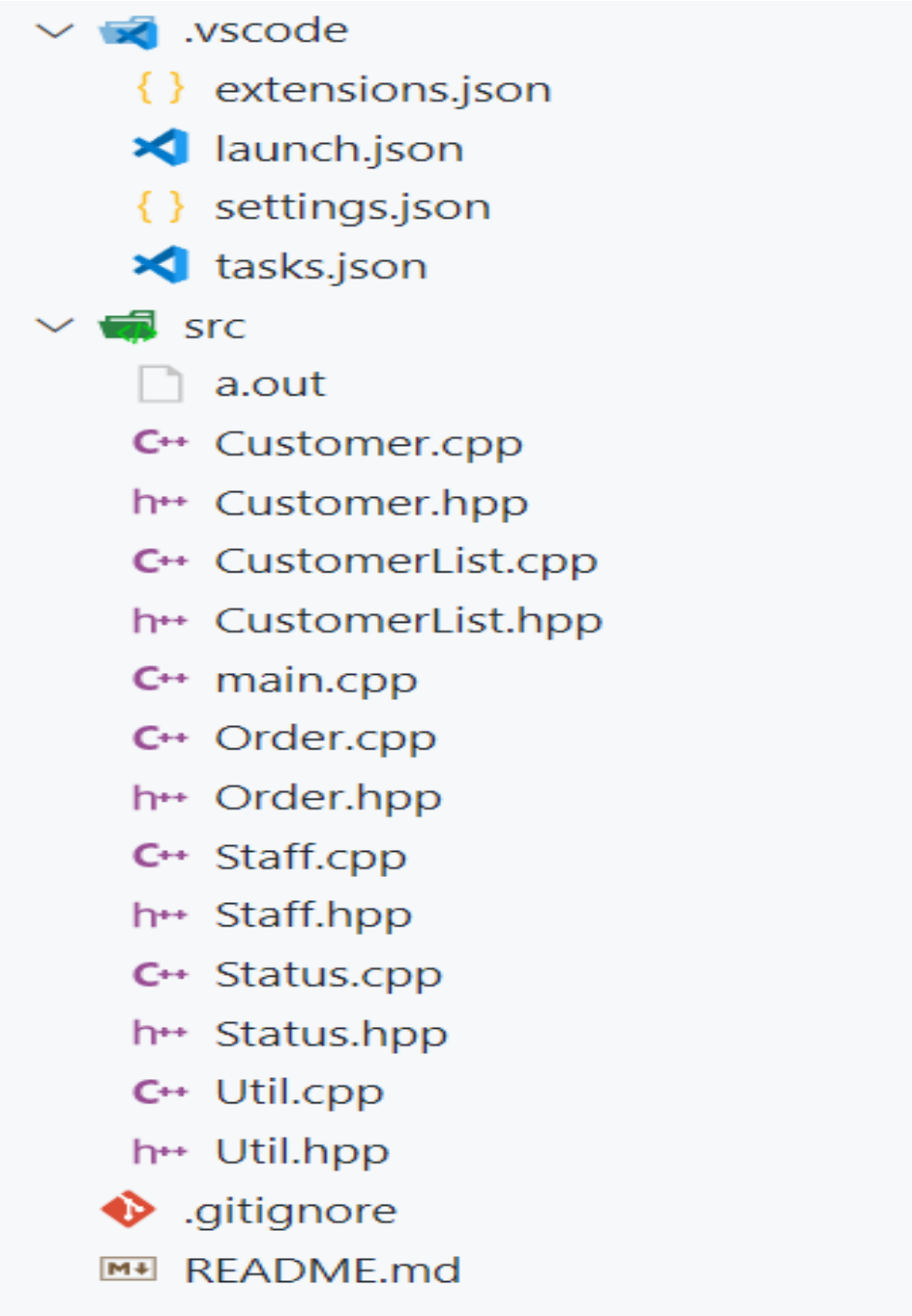
Order	
Responsibility	**Collaborators**
Store Order details Place Order Assign staff members for each order Calculate Total Charge per Order Display Order summery	Customer Staff Util

Staff	
Responsibility	**Collaborators**
Input employee details Process the refund	

Status	
Responsibility	**Collaborators**
Update order status.	Order Staff Util

Util	
Responsibility	**Collaborators**
Generate universally unique identifiers	

Project Structure



Individual Contributions

Registration No	Name	Contribution
All members		main.cpp
IT21258176	Sanuja Methmal	Customer.hpp Customer.cpp CustomerList.hpp CustomerList.cpp
IT21337512	Thisun Senaratna	Order.hpp Order.cpp
IT21266232	Induwara Ashinshana	Staff.hpp Staff.cpp
IT21283376	Avishka Pramod	Util.hpp Util.cpp
IT21245756	Pivithuru Milan	Status.hpp Status.cpp

GitHub repository: IT1050-2022-Feb/ooc-project-IT21258176: ooc-project-IT21258176 created by GitHub Classroom

Source Code

main.cpp

by all members

```
#include <iostream>
#include "Customer.hpp"
#include "Order.hpp"
#include "Status.hpp"
#include "Staff.hpp"
#include "CustomerList.hpp"

using namespace std;

int main(int argc, char** argv) {
  //Create new customers
  Customer* methmal = new Customer("methmal", "943hfg9HFIK94HW2");
  methmal
    ->addEmail("methmal48y8@gmail.com")
    ->addEmail("methmal768@yahoo.com")
    ->addContactNo("0772312345")
    ->addContactNo("0761234322")
    ->addLocation("Ragama, Jaela");

  Customer* namal = new Customer("namal", "dfjkl546dsf");
  namal
    ->addEmail("Kkamal@gmail.com")
    ->addContactNo("0776123445")
    ->addContactNo("0712354667")
    ->addLocation("Gampaha, Ganemulla")
    ->addLocation("Hapugoda, Kandana");

  CustomerList* list;
  list->push(methmal)->push(namal);


  //edit customer profile
  Customer* loggedCustomer = list->login("methmal", "943hfg9HFIK94HW");
  loggedCustomer
    ->removeEmail("methmal48y8@gmail.com")
    ->removeContactNo("0772312345")
    ->addContactNo("0741234329")
    ->changePassword("943hfg9HFIK94HW", "8rhf898939gHFU");
```

```
    //create staff members
    Staff* s1 = new Staff();
    s1->setName("Amara");
    s1->setJob("Driver");

    Staff* s2 = new Staff();
    s2->setName("Kamala");
    s2->setJob("Cleaner");


    //namal place an order
    loggedCustomer = list->login("namal", "dfjkl546ds");

    Order* o1 = new Order('d', 3.58);
    o1->displayOrderDetails();
    o1->assignStaff(s1);
    o1->placedBy(loggedCustomer);

    //update order status
    Status* st1 = new Status("Picked");
    st1->assignOrder(o1);
    st1->assignStaff(s1);


    //methmal place an order
    loggedCustomer = list->login("methmal", "9rhf898939gHFU");

    Order* o2 = new Order('l', 9.54);
    o2->displayOrderDetails();
    o2->assignStaff(s2);
    o2->placedBy(loggedCustomer);

    //update order status
    Status st2("Packed");
    st2.assignOrder(o2);
    st2.assignStaff(s2);


    //clean memory
    delete s1, s2;
    delete o1, o2;
    delete st1, st2;
    delete methmal, namal;
    delete list;
}
```

Customer.hpp

by Sanuja Methmal | IT21258176

```
#ifndef CUSTOMER HPP
#define CUSTOMER HPP

#include <iostream>
#include <vector>

using std::string;
using std::vector;

class Customer {
 private:
  string id, username, password;
  vector<string> emails, contactNos, locations;

 public:
  Customer(string _username, string _password);
  string getId();
  bool isUsernameEqual(string);
  bool isPasswordEqual(string);
  Customer* changeUsername(string);
  Customer* changePassword(string oldPassword, string newPassword);
  Customer* addEmail(string);
  Customer* removeEmail(string);
  Customer* addContactNo(string);
  Customer* removeContactNo(string);
  Customer* addLocation(string);
  Customer* removeLocation(string);
};

#endif
```

Customer.cpp

by Sanuja Methmal | IT21258176

```
#include "Customer.hpp"
#include <string.h>
#include <iostream>
#include "Util.hpp"

using std::cout;
using std::endl;
using std::string;

Customer::Customer(string  username, string  password) {
  Util u;
  id = u.generateRandomId();
  username =  username;
  password =  password;
}

string Customer::getId() {
  return id;
}

bool Customer::isUsernameEqual(string _username) {
  return strcmp(username.c_str(),  username.c_str()) == 0;
}

Customer* Customer::changeUsername(string  username) {
  username =  username;
  return this;
}

Customer* Customer::changePassword(string oldPassword, string newPassword) {
  if (password != oldPassword) {
    cout << "Incorrect password";
    exit(EXIT_FAILURE);
  }
  password = newPassword;
  return this;
}

bool Customer::isPasswordEqual(string  password) {
  bool match = password ==  password;
  return match;
}
```

```
Customer* Customer::addEmail(string email) {
  emails.push back(email);
  return this;
}

Customer* Customer::removeEmail(string  email) {
  for (int i = 0; i < emails.size(); i++) {
    if (strcmp(emails[i].c str(),  email.c str()))
      continue;
    emails.erase(emails.begin() + i);
    return this;
  }
  return this;
}

Customer* Customer::addContactNo(string contactNo) {
  contactNos.push back(contactNo);
  return this;
}

Customer* Customer::removeContactNo(string no) {
  for (int i = 0; i < contactNos.size(); i++) {
    if (strcmp(contactNos[i].c str(), no.c str()))
      continue;
    contactNos.erase(contactNos.begin() + i);
    return this;
  }
  return this;
}

Customer* Customer::addLocation(string location) {
  locations.push back(location);
  return this;
}

Customer* Customer::removeLocation(string location) {
  for (int i = 0; i < locations.size(); i++) {
    if (strcmp(locations[i].c str(), location.c str()))
      continue;
    locations.erase(locations.begin() + i);
    return this;
  }
  return this;
}
```

CustomerList.hpp

by Sanuja Methmal | IT21258176

```
#ifndef CUSTOMER_LIST_HPP
#define CUSTOMER_LIST_HPP

#include <vector>
#include "Customer.hpp"

using std::vector;

class CustomerList {
 private:
  vector<Customer*> elements;

 public:
  CustomerList();
  CustomerList* push(Customer*);
  Customer* login(string username, string password);
};

#endif
```

CustomerList.cpp

by Sanuja Methmal | IT21258176

```
##include "CustomerList.hpp"

#include <string.h>
#include <iostream>
#include "Customer.hpp"

using namespace std;

CustomerList* CustomerList::push(Customer* cus) {
  elements.push back(cus);
  return this;
}

Customer* CustomerList::login(string username, string password) {
  for (Customer* cus : elements) {
    if (!cus->isUsernameEqual(username))
      continue;
    if (cus->isPasswordEqual(password))
      return cus;
    cout << "Incorrect password" << endl;
    exit(EXIT_FAILURE);
  }
  cout << "User not found!" << endl;
  exit(EXIT FAILURE);
}
```

Order.hpp

by Thisun Senaratna | IT21337512

```
#ifndef Order_HPP
#define Order_HPP
#include "Customer.hpp"
#include "Staff.hpp"
#include <iostream>

using namespace std;

class Order {
private:
    string id;
    char type;
    float weight;
    Customer* placedCustomer;
    Staff* assignedStaff;
    float calculateSubtotal();
    float calculateTax();

public:
    Order(char, float);
    void placedBy(Customer*);
    float calcualteTotal();
    string getId();
    void displayOrderDetails();
    void assignStaff(Staff*);
};

#endif
```

Order.cpp

by Thisun Senaratna | IT21337512

```cpp
#include <stdio.h>
#include <iostream>
#include "Staff.hpp"
#include "Order.hpp"
#include "Util.hpp"
using namespace std;

Order::Order(char  type, float  weight) {
  Util u;
  id = u.generateRandomId();
  type =  type;
  weight =  weight;
}

void Order::placedBy(Customer* c) {
  placedCustomer = c;
}

float Order::calculateSubtotal() {
  float ratePerKg = (type == 'd') ? 600 : 350;  // ternary operator
  return weight * ratePerKg;
}

float Order::calculateTax() {
  float TAX RATE = (8.0 / 100), tax;
  tax = calculateSubtotal() * TAX RATE;
  return tax;
}

float Order::calcualteTotal() {
  float total;
  total = calculateSubtotal() + calculateTax();
  return total;
}

string Order::getId() {
  return id;
}
```

```
void Order::displayOrderDetails() {
  cout << "Order ID: " << id << endl;
  if (type = 'd')
    cout << "Order Type: Dryclean" << endl;
  else
    cout << "Order Type: Laundry" << endl;

  printf("Weight: %.2f/n", weight);
  cout << "Sub Total" << calculateSubtotal() << endl
    << "Tax: " << calculateTax() << endl
    << " Total Charge: " << calcualteTotal() << endl;
}

void Order::assignStaff(Staff* s) {
  assignedStaff = s;
}
```

Staff.hpp

by Induwara Ashinshna | IT21266232

```
#ifndef Staff_HPP
#define Staff_HPP

#include <iostream>
using namespace std;

class Staff {
private:
    string id;
    string jobtitle;
    string name;

public:
    Staff();
    string getId();
    void setName(string Name);
    void setJob(string Job);
};

#endif
```

Staff.cpp

by Induwara Ashinshna | IT21266232

```cpp
#include "Staff.hpp"
#include "Util.hpp"
#include <iostream>

using namespace std;

Staff::Staff() {
    Util x;
    id = x.generateRandomId();
}

void Staff::setName(string Name) {
    name = Name;

}
void Staff::setJob(string Job) {
    jobtitle = Job;
}

string Staff::getId() {
    return id;
}
```

Status.hpp

by Pivithuru Milan | IT21245756

```
#ifndef Status_HPP
#define Status_HPP

#include <ctime>
#include <iostream>
#include "Order.hpp"
#include "Staff.hpp"
using namespace std;

class Status {
 private:
  string id;
  string code;
  string datetime;
  Staff* staff;
  Order* order;

 public:
  Status(string);
  void assignStaff(Staff*);
  void assignOrder(Order*);
};
#endif
```

Status.cpp

by Pivithuru Milan | IT21245756

```cpp
#include "Status.hpp"
#include <ctime>
#include <iostream>
#include "Order.hpp"
#include "Staff.hpp"
using namespace std;

Status::Status(string _code) {
  time_t now = time(0);
  datetime = ctime(&now);
  code = _code;
}

void Status::assignStaff(Staff* s) {
  staff = s;
}

void Status::assignOrder(Order* o) {
  order = o;
}
```

Util.hpp

by Avishka Pramod | IT21283376

```
#ifndef UTIL_HPP
#define UTIL_HPP

#include <iostream>

using namespace std;

class Util {
public:
    char getRandomChar();
    string generateRandomId();
    long hash(char*);
};

#endif
```

Util.cpp

by Avishka Pramod | IT21283376

```
#include "Util.hpp"
#include <algorithm>
#include <climits>
#include <functional>
#include <iostream>
#include <random>
#include <sstream>
#include <vector>

using namespace std;

char Util::getRandomChar() {
  random_device rd;
  mt19937 gen(rd());
  uniform_int_distribution<> dis(0, 255);
  return static_cast<char>(dis(gen));
}

string Util::generateRandomId() {
  const int LENGTH = 10;
  stringstream ss;

  for (auto i = 0; i < LENGTH; i++) {
    char rc = getRandomChar();
    stringstream hexstream;
    hexstream << hex << int(rc);
    string hex = hexstream.str();
    ss << (hex.length() < 2 ? '0' + hex : hex);
  }

  return ss.str();
}
```