# Design Document and Test Plan

Name of team members who collaborated on the design and test plan:

1. Name (*first last*): Angie Diaz
2. Name (*first last*): Janaki Bhosale
3. Name (*first last*): Dylan Harvey

Name of programming for which you submit this document: **Parking Permits**

UML Class Diagram

## Motorcycles: Dylan

-make: string
-model: string
-cc: int
-capacity: int
-year: int

+Motorcycles()
+Motorcycles(m:string, o:string, c:int, p:int, y:int)
+validateInput(v:string): bool
+validateInput(v:int): bool
+setMake(m:string): bool
+setModel(o:string): bool
+setCC(c:int): bool
+setCapacity(p:int): bool
+setYear(y:int): bool
+getMake(): string
+getModel(): string
+getCC(): int
+getCapacity(): int
+getYear(): int

## LowEmissions: Angie

-make: string
-model: string
-weight: int
-mpg: int
-year: int

+LowEmissions()
+LowEmissions(m:string, o:string, w:int, p:int, y:int)
+validateInput(v:string): bool
+validateInput(v:int): bool
+setMake(m:string): bool
+setModel(o:string): bool
+setWeight(w:int): bool
+setMilesPerGal(p:int): bool
+setYear(y:int): bool
+getMake(): string
+getModel(): string
+getWeight(): int
+getMPG(): int
+getYear(): int

## Invoice: Dylan

-permitPrice: double
-serviceCharge: double
-discount: double

+Invoice()
+Invoice(p:double, s:double, d:double)
+validateInput(v:double): bool
+setPermitPrice(p:double): bool
+setServiceCharge(s:double): bool
+setDiscount(d:double): bool
+getPermitPrice(): double
+getServiceCharge(): double
+getDiscount(): double
+calcTotal() const: double
+printInvoice(const CustomerType& customer, const VehicleType& vehicle) const: string

## Pseudocode

### In main.cpp:

- Include header files and directives
- Initialize customerType, name, email, address, make, model string variables
- Initialize year integer variable, serviceCharge, discount to 0.0, double variable, serviceFee to 25.00 (double variable)
- Ask user to enter if they are an employee, student, or visitor, store in customerType
- In a while loop, continue to ask if customerType does not match either of the options
- If customerType is equal to employee
    - initialize employeeID and yearsEmployed variable
    - Ask user to enter name, email, address, employeeID, and years employed and store into corresponding variables
    - Create Employee object sending the variables name, email, address, employeeID and yearsEmployed as arguments
- Else if customerType is equal to student
    - Initialize studentID and level variables
    - Ask user to enter name, email, address, studentID, and education level and store into corresponding variables
    - Create Student object sending the variables name, email, address, studentID and level as arguments
- Else
    - Initialize registrationNum and firstTime variables
    - Ask if user is visiting for the first time
        - If yes
            - set firstTime equal to true
            - set discount equal to 5.00
        - Else, set firstTime equal to false
    - Ask user to enter name, email, address, and registrationNum
    - Create Visitor object sending name, email, address, registrationNum, and firstTime as arguments
- Initialize vehicleChoice variable
- Ask user to select between regular, motorcycle, or low emission vehicle and store in vehicleChoice
- In a while loop, continue to ask for vehicleChoice until it corresponds to either of the options
- If vehicleChoice is equal to regular
    - Initialize color and licensePlate string variables
    - Ask user to enter make, model, year, color, and license plate, and store in corresponding variables
    - Create regular class object and send make, model, year, color and licensePlate as arguments

- Else if vehicleChoice is equal to motorcycle
    - Initialize CC and capacity int variables
    - Ask user to enter make, model, year, capacity, and CC and store in respective variables
    - Create motorcycle class object and send make, model, year, capacity, and CC as arguments
- Else
    - Initialize milesPerGal and weight variables
    - Ask user to enter make, model, year, miles per gallon, and weight of vehicle and store in corresponding variables
    - Create lowEmissions object and send make, model, year, miles per gallon, and weight as arguments
- Initialize permitPrice double variable to the price per day (5 dollars)
- Initialize permitChoice variable
- Ask user to choose between annual, semester and one-day permits
- While permitChoice does not equal annual, semester or one-day continue to ask
- If permitChoice is equal to annual:
    - Multiply permitPrice by days in an academic year
    - Create Invoice object, send permitPrice, serviceCharge, and discount variables as objects
    - In a print statement, call on the Invoice object printInvoice() function and send customer and vehicle objects
        - If customerType==student
            - if vehicleChoice==regular
                - Send printInvoice() with student and regular objects
            - else if vehicleChoice==motorcycle
                - Send printInvoice() with student and motorcycle objects
            - Else
                - Send printInvoice() with student and lowEmission objects
        - Else if customerType==Employee
            - if vehicleChoice==regular
                - Send printInvoice() with employee and regular objects
            - else if vehicleChoice==motorcycle
                - Send printInvoice() with employee and motorcycle objects
            - Else
                - Send printInvoice() with employee and lowEmission objects
        - Else
            - if vehicleChoice==regular
                - Send printInvoice() with visitor and regular objects
            - else if vehicleChoice==motorcycle
                - Send printInvoice() with visitor and motorcycle objects
            - Else
                - Send printInvoice() with visitor and lowEmission objects

- Else if permitChoice is equal to semester:
    - Multiply permitPrice by days in an academic semester
    - Create Invoice object, send permitPrice, serviceCharge, and discount variables as objects
    - In a print statement, call on the Invoice object printInvoice() function and send customer and vehicle objects
        - If customerType==student
            - if vehicleChoice==regular
                - Send printInvoice() with student and regular objects
            - else if vehicleChoice==motorcycle
                - Send printInvoice() with student and motorcycle objects
            - Else
                - Send printInvoice() with student and lowEmission objects
        - Else if customerType==Employee
            - if vehicleChoice==regular
                - Send printInvoice() with employee and regular objects
            - else if vehicleChoice==motorcycle
                - Send printInvoice() with employee and motorcycle objects
            - Else
                - Send printInvoice() with employee and lowEmission objects
        - Else
            - if vehicleChoice==regular
                - Send printInvoice() with visitor and regular objects
            - else if vehicleChoice==motorcycle
                - Send printInvoice() with visitor and motorcycle objects
            - Else
                - Send printInvoice() with visitor and lowEmission objects
- Else
    - Create Invoice object and send permitPrice, serviceCharge, and discount variables as objects
    - In a print statement, call on the Invoice object printInvoice() function and send customer and vehicle objects
        - If customerType==student
            - if vehicleChoice==regular
                - Send printInvoice() with student and regular objects
            - else if vehicleChoice==motorcycle
                - Send printInvoice() with student and motorcycle objects
            - Else
                - Send printInvoice() with student and lowEmission objects
        - Else if customerType==Employee
            - if vehicleChoice==regular
                - Send printInvoice() with employee and regular objects

- else if vehicleChoice==motorcycle
  - Send printInvoice() with employee and motorcycle objects
- Else
  - Send printInvoice() with employee and lowEmission objects

■ Else

- if vehicleChoice==regular
  - Send printInvoice() with visitor and regular objects
- else if vehicleChoice==motorcycle
  - Send printInvoice() with visitor and motorcycle objects
- Else
  - Send printInvoice() with visitor and lowEmission objects

- End program

## In Visitors.h:

- Header Guards
- Include string
- Using namespace std
- Define class Visitors (from UML Diagram)
- Private:
  - string name
  - string email
  - string address
  - int regNumber
  - bool firstVisit
- Public:
  - Visitors() - Default constructor
  - Visitors(string n, string e, string a, int rN, bool fV)
  - bool validateInput(string v)
  - bool validateInput(int v)
  - bool validateInput(bool v)
  - bool setName(string n)
  - bool setEmail(string e)
  - bool setAddress(string a)
  - bool setRegNumber(int rN)
  - bool setFirstVisit(bool fV)
  - string getName()
  - string getEmail()
  - string getAddress()
  - int getRegNumber()
  - bool getFirstVisit()

## In Visitors.cpp:

- Include Visitors.h
- bool Visitors::validateInput(string v)
    - if (v.empty())
        - return 0;
    - else
        - return 1;
- bool Visitors::validateInput(int v)
    - if (v < 0)
        - return 0;
    - else
        - return 1;
- bool Visitors::validateInput(bool v)
    - if (v != 0 && v != 1)
        - return 0;
    - else
        - return 1;
- bool Visitors::setName(string n)
    - if (validateInput(n))
        - name = n;
        - return 1;
    - else
        - return 0;
- bool Visitors::setEmail(string e)
    - if (validateInput(e))
        - email = e;
        - return 1;
    - else
        - return 0;
- bool Visitors::setAddress(string a)
    - if (validateInput(a))
        - address = a;
        - return 1;
    - else
        - return 0;
- bool Visitors::setRegNumber(int rN)
    - if (validateInput(rN))
        - regNumber = rN;
        - return 1;
    - else
        - return 0;

- bool Visitors::setFirstVisit(bool fV)
    - if (validateInput(fV))
        - firstVisit = fV;
        - return 1;
    - else
        - return 0;
- string Visitors::getName()
    - return name;
- string Visitors::getEmail()
    - return email;
- string Visitors::getAddress()
    - return address;
- int Visitors::getRegNumber()
    - return regNumber;
- bool Visitors::getFirstVisit()
    - return firstVisit;

## In Students.h:

- Header Guards
- Include string
- Using namespace std
- Define class students (from UML diagram)
- Private:
    - string name
    - string email
    - string address
    - int studentID
    - int educationLevel
- Public:
    - Students() - default constructor
    - Students(string n, string e, string a, int sID, int l)
    - bool validateInput(string v)
    - bool validateInput(int v)
    - bool validateInput(bool v)
    - bool setName(string n)
    - bool setEmail(string e)
    - bool setAddress(string a)
    - bool setStudetID(int d)
    - void setLevel(int l)
    - string getName()
    - string getEmail()
    - string getAddress()

    - int getStudentID()
    - int getLevel()
- End Header Guards

## In Students.cpp:

- Include Students.h
- bool Students::validateInput(string v)
    - if (v.empty())
        - return 0;
    - else
        - return 1;
- bool Students::validateInput(int v)
    - if (v < 0)
        - return 0;
    - else
        - return 1;
- bool Students::validateInput(bool v)
    - if (v != 0 && v != 1)
        - return 0;
    - else
        - return 1;
- bool Students::setName(string n)
    - if (validateInput(n))
        - name = n;
        - return 1;
    - else
        - return 0;
- bool Students::setEmail(string e)
    - if (validateInput(e))
        - email = e;
        - return 1;
    - else
        - return 0;
- bool Students::setAddress(string a)
    - if (validateInput(a))
        - address = a;
        - return 1;
    - else
        - return 0;
- bool Students:: setStudentID(int d)
    - if (validateInput(d))
        - studentID = d;

            - return 1;
        - else
            - return 0;
- bool Employees:: setLevel(int y)
    - if (validateInput(y))
        - level = l;
        - return 1;
    - else
        - return 0;
- string Students:getName()
    - return name;
- string Students::getEmail()
    - return email;
- string Students::getAddress()
    - return address;
- int Students::getStudentID()
    - return studentID;
- int Students::getLevel()
    - return level;

## In Employees.h

- Header guards
- Include string
- Using namespace std
- Define class Employees (UML Diagram)
- Private:
    - string name
    - string email
    - int employeeID
    - int yearsEmployed
- Public:
    - Employees(): default constructor
    - Employees(string n, string e, string a, int eID, int yE)
    - bool validateInput (int v)
    - bool setName(string n)
    - bool setEmail(string e)
    - bool setAddress(string a)
    - bool setEmployeeID(int d)
    - bool setYearsEmployed(int y)
    - string getName()
    - string getEmaill()
    - string getEmployeeID()

- int getYearsEmployed()

## In Employee.cpp

- Include Employees.h
- bool Employees::validateInput(string v)
  - if (v.empty())
    - return 0;
  - else
    - return 1;
- bool Employees::validateInput(int v)
  - if (v < 0)
    - return 0;
  - else
    - return 1;
- bool Employees::setName(string n)
  - if (validateInput(n))
    - name = n;
    - return 1;
  - else
    - return 0;
- bool Employees::setEmail(string e)
  - if (validateInput(e))
    - email = e;
    - return 1;
  - else
    - return 0;
- bool Employees::setAddress(string a)
  - if (validateInput(a))
    - address = a;
    - return 1;
  - else
    - return 0;
- bool Employees:: setEmployeeID(int d)
  - if (validateInput(d))
    - employeeID = d;
    - return 1;
  - else
    - return 0;
- bool Employees:: setYearsEmployed(int y)
  - if (validateInput(y))
    - yearsEmployed = y;
    - return 1;

        - else
            - return 0;
- string Employees:getName()
    - return name;
- string Employees::getEmail()
    - return email;
- string Employees::getAddress()
    - return address;
- int Employees::getEmployeeID()
    - return employeeID;
- int Employees::getYearsEmployed()
    - return yearsEmployed;

## In Motorcycles.h:

- Header Guards
- Include string
- Using namespace std
- Define class Motorcycles (from UML diagram)
- Private:
    - string make
    - string model
    - int capacity
    - int cc
    - int year
- Public:
    - Motorcycles() - default constructor
    - Motorcycles(string m, string o, int c, int p, int y)
    - bool validateInput(string v)
    - bool validateInput(int v)
    - bool setMake(string m)
    - bool setModel(string o)
    - bool setCC(int c)
    - bool setCapacity(int p)
    - bool setYear(int y)
    - string getMake()
    - string getModel()
    - string getColor()
    - int getYear()

## In Motorcycles.cpp:

- Include Motorcycles.h
- bool Motorcycles::validateInput(string v)
    - if (v.empty())
        - return 0;
    - else
        - return 1;
- bool Motorcycles::validateInput(int v)
    - if (v < 0)
        - return 0;
    - else
        - return 1;
- bool Motorcycles::setMake(string m)
    - if (validateInput(m))
        - make = m;
        - return 1;
    - else
        - return 0;
- bool Motorcycles::setModel(string o)
    - if (validateInput(o))
        - model = o;
        - return 1;
    - else
        - return 0;
- bool Motorcycles::setCC(int c)
    - if (validateInput(c))
        - cc = c;
        - return 1;
    - else
        - return 0;
- bool Motorcycles::setCapacity(int p)
    - if (validateInput(p))
        - capacity = p;
        - return 1;
    - else
        - return 0;
- bool Motorcycles::setYear(int y)
    - if (validateInput(y))
        - year = y;
        - return 1;
    - else

        - return 0;
- string Motorcycles::getMake()
    - return make;
- string Motorcycles::getModel()
    - return model;
- int Motorcycles::getCC()
    - return cc;
- int Motorcycles::getCapacity()
    - return capacity;
- int Motorcycles::getYear()
    - return year;

## In LowEmissions.h:

- Include header files
- Include string
- Using namespace std
- Define class LowEmissions (from UML diagram)
- Private:
    - string make
    - string model
    - int weight
    - int mpg
    - int year
- Public:
    - LowEmissions() - default constructor
    - LowEmissions(string m, string o, int w, int p, int y)
    - bool validateInput(string v)
    - bool validateInput(int v)
    - bool setMake(string m)
    - bool setModel(string o)
    - bool setWeight(int w)
    - bool MilesPerGal(int p)
    - bool setYear(int y)
    - string getMake()
    - string getModel()
    - int getWeight()
    - int  getMilesPerGal()
    - int getYear()

## In LowEmissions.cpp:

- Include LowEmissions.h
- bool LowEmissions::validateInput(string v)
    - if (v.empty())
        - return 0;
    - else
        - return 1;
- bool LowEmissions::validateInput(int v)
    - if (v < 0)
        - return 0;
    - else
        - return 1;
- bool LowEmissions::setMake(string m)
    - if (validateInput(m))
        - make = m;
        - return 1;
    - else
        - return 0;
- bool LowEmissions::setModel(string o)
    - if (validateInput(o))
        - model = o;
        - return 1;
    - else
        - return 0;
- bool LowEmissions::setWeight(int w)
    - if (validateInput(w))
        - weight= w;
        - return 1;
    - else
        - return 0;
- bool LowEmissions::setMilesPerGal(int p)
    - if (validateInput(p))
        - mpg = p;
        - return 1;
    - else
        - return 0;
- bool Motorcycles::setYear(int y)
    - if (validateInput(y))
        - year = y;
        - return 1;
    - else

        - return 0;
- string LowEmissions::getMake()
    - return make;
- string LowEmissions::getModel()
    - return model;
- int LowEmissions::getWeight()
    - return weight;
- int LowEmissions::getMilesPerGal()
    - return mpg;
- int LowEmissions::getYear()
    - return year;

## In Invoice.h:

- Header Guards
- Include String
- Include all the other class header files
- Use namespace std
- Define class Invoice (From UML Diagram)
- Private:
    - double permitPrice
    - double serviceCharge
    - double discount
- Public:
    - Invoice()
    - Invoice(double p, double s, double d)
    - bool validateInput(double v)
    - bool setPermitPrice(double p)
    - bool setServiceCharge(double s)
    - bool setDiscount(double d)
    - double getPermitPrice()
    - double getServiceCharge()
    - double getDiscount()
    - double calcTotal()
    - string printInvoice(const CustomerType& customer, const VehicleType& vehicle) const

## In Invoice.cpp:

- Include Invoice.h
- bool Invoice::validateInput(double v)
    - if (v < 0)
        - return 0;
    - else
        - return 1;
- bool Invoice::setPermitPrice(double p)
    - if (validateInput(p))
        - permitPrice = p;
        - return 1;
    - else
        - return 0;
- bool Invoice::setServiceCharge(double s)
    - if (validateInput(s))
        - serviceCharge = s;
        - return 1;
    - else
        - return 0;
- bool Invoice::setDiscount(double d)
    - if (validateInput(d))
        - discount = d;
        - return 1;
    - else
        - return 0;
- double Invoice::getPermitPrice()
    - return permitPrice;
- double Invoice::getServiceCharge()
    - return serviceCharge;
- double Invoice::getDiscount()
    - return discount;
- double Invoice::calcTotal() const
    - return (permitPrice + serviceCharge - discount);
- template<typename CustomerType, typename VehicleType>
  string Invoice::printInvoice(const CustomerType& customer, const VehicleType& vehicle) const
    - stringstream invoiceDetails;
    - Customer info
    - Vehicle info
    - Price details
    - return invoiceDetails.str();

# Test Plan

*(See Ch. 5.13 in our textbook for an example of how to write a test plan)*

| Test # | Purpose | Input | Expected Output |
|---|---|---|---|
| 1 | Test correct input with visitor, motorcycle, and one time permit. (First time visitor, makes it free for the day since 5-5=0) | visitor, yes, Joe, joe@gmail.com, 123 real street, 47321, motorcycle, Honda, grom, 2015, 1, 125, one-day. | **Customer Information:**<br>Joe<br>Joe@gmail.com<br>123 real street<br>Registration Number: 47321<br>First Visit: Yes<br>**Vehicle Information:**<br>Make: Honda<br>Model: grom<br>Year: 2015<br>CCs: 125<br>Capacity: 1<br>**Price Details:**<br>Permit Price: $5.00<br>Service Charge: $0.00<br>Discount: $5.00<br>**Total: $0.00** |
| 2 | Test correct input with Employee, LowEmissions, and one time permit | employee, John, johndoe@clemson.edu, 555 Real Road, 2343456, 20, low emission, Tesla, Model 3, 4000, 134, 2017, one-day | **Customer Information:**<br>John<br>johndoe@clemson.edu<br>555 Real Road<br>EmployeeID: 2343456<br>Years Employed: 20<br>**Vehicle Information:**<br>Make: Tesla<br>Model: Model 3<br>Weight (to the nearest lb): 4000<br>MPG: 134<br>Year: 2017<br>**Price Details:**<br>Permit Price: $5.00<br>Service Charge: $5.00<br>Discount: $0.00<br>Total: $10.00 |
| 3 | Test correct input with Employee, LowEmissions, and semester permit | employee, Jane, janedoe@clemson.edu, 555 Real Road, 3343456, 5, low emission, Tesla, | **Customer Information:**<br>Jane<br>janedoe@clemson.edu<br>555 Real Road<br>EmployeeID: 3343456 |

| | | | |
|---|---|---|---|
| | | Model 3, 4000, 134, 2020, semester | Years Employed: 5<br>**Vehicle Information:**<br>Make: Tesla<br>Model: Model 3<br>Weight (to the nearest lb): 4000<br>MPG: 134<br>Year: 2020<br>**Price Details:**<br>Permit Price: $525.00<br>Service Charge: $5.00<br>Discount: $0.00<br>Total: $530.00 |
| 4 | Test correct input with student, regular vehicle, and semester permit. | student, no, Jack, jack@gmail.com, 123 real drive, 13579, senior, Toyota, Camry, black, abc123, 2020, semester. | **Customer Information:**<br>Jack<br>jack@gmail.com<br>123 real drive<br>studentID: 13579<br>First Visit: no<br>**Vehicle Information:**<br>Make: Toyota<br>Model: Camry<br>Year: 2020<br>Color: black<br>License Plate: abc123<br>**Price Details:**<br>Permit Price: $525.00<br>Service Charge: $5.00<br>Discount: $0.00<br>Total: $530.00 |
| 5 | Test various incorrect inputs (customer type, values, vehicle type, etc.) | e, , ceo, student, … (continues)<br>car, , e, regular, … (continues)<br>, e, student, semester | Correct invoice with respective data, main point is to ensure the correct logic is chosen and the data types are correct. |
| 6 | Test incorrect data inputs | visitor, no, John, e@gmail.com, 234 fake street, -1, regular, Toyota, prius, -2023, 4, gray, abc123, one-day. | Correct invoice with respective data, main point is to ensure the incorrect values were not taken, and replaced with default values. |

Note: for inputs from a list of choices/menu may vary slightly (could use numbered lists or strings for the names, but we'll have to modify slightly at implementation)