Winter Semester 2022

CSE250 Database Management System

Project title:

**“HOME MAINTENANCE SERVICES”** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Team Members:

|  |  |
| --- | --- |
| Name | Enrollment Number |
| Akshat Shah | AU2040052 |
| Omkar Pandya | AU2040180 |
| Kinal Kagathara | AU2040240 |



ABSTRACT

In olden days, there were no resources for getting services for home. At that time, we had to ask people for the contact number or the shop address and then we had to meet them in person. Then they were supposed to come to our place and do the needful. This process takes a long time and lots of effort. But, in this technology driven world, we need the work to be done at the maximum speed possible. So we came up with an idea to develop this portal to provide the basic home maintenance services at ease.

So, by using this home maintenance service website, users can get reliable service at their own convenient time and from any place. Services provided by the system are Pest-control, Appliance repairing, Home cleaning, Medical, Computer repairing, Painting, Carpentry, Plumbing, Mechanical, Electrical, Packers & Movers etc. The skilled service person will visit the place for which, visiting charges will be pre informed. The charges of material will be informed on site.

In our system, the customer can register using their email and contact number. They can request for a specific service by clicking on that service and provide the necessary details like residential address, contact-number, area pin-code, e-mail id. Here when customers register for service. Service will be provided as per the time scheduled by the expert who will be going to provide that service.

In the proposed system, customers have to make the payment by credit card or debit card. If due to any unavoidable reasons, if any of the users need to cancel the service, then we have provisions to return 50% of the visiting charges. There is also provision for a person to register for providing the services. Those who are interested in providing the services can register by providing the digital copy of the following documents:

* Passport photo
* Police verification certificate
* Residential proof

The front end of the project is developed in PHP (HyperText Preprocessor) language.

**1. INTRODUCTION**

**1.1 CHARACTERISTICS OF EXISTING SYSTEM**

* In the manual system, the contact number or address of the individual’s shop has to be taken from the people and after that, they are needed to be contacted either by calling them or meeting personally.

**1.2 OVERVIEW OF PROPOSED SYSTEM**

* Provides the information of the person going to provide service (Experience, Background)
* The person who wants to provide service can also join
* Client can also book for AMC(Annual Maintenance Contract)
* Client can also select emergency service on emergency time
* Feedback
* Provide skilled service men for any Problem
* Tips for Maintenance

**1.3 SCOPE**

* In our proposed system the client can book services by viewing all the services and its sub services.
* The client can choose a particular service person.
* Clients can book AMC (Annual Maintenance Contract) for any services.
* Service providers can register for providing services in their area of expertise.
* Clients can get tips for the services.
* Clients can cancel service before the 2 hour of their service providing time.
* Clients can also give feedback to the service man.
* Clients can also see the feedback of servicemen provided by other clients and their work experience.

**2. SOFTWARE REQUIREMENT SPECIFICATION**

**2.1 USER CHARACTERISTICS**

**1. ADMIN**

* All the authorities lie with the admin.
* Admin can manage the feedback.
* Admin have rights to
  + Add or Remove any service.
  + Delete account of any member.
  + Decline the service provider.

**2. CLIENT**

* Clients have to register and then login and book services.
* Clients can cancel services.
* Clients can provide their feedback on servicemen.

**3. SERVICE PROVIDER**

* Person who wants to provide service can register for the same by providing some details.
* They are supposed to be informed about their appointment via email.

**2.2 System Requirements**

* PHP 5.5.11
* SQL 5.6.16
* XAMPP must be installed and the database must be imported to XAMPP

**2.3 FUNCTIONAL REQUIREMENTS**

**1. Register for account**

INPUT: Enter all the required detail for registration

PROCESS: System checks that particular Email-Id exist or not if not exist then store into database

OUTPUT: “Registered successfully” message display and open Home page

**2. Login**

INPUT: Enter username or password

PROCESS: Check from database username exists or not

OUTPUT: “Login successfully” message display and open Home page

**3. Book services**

INPUT: Select services, related sub service and service provider

PROCESS: Service details is stored in database and inform the service provider

OUTPUT: “Your service is Booked” and client get time for service

**4. Cancel services**

INPUT: Select Service, Sub Service, Service Provider Name and date

PROCESS: Go to Database and cancel the Entry for that Service

OUTPUT: “Cancel Service” Message on Screen

**5. Book AMC(Annual Maintenance Contract)**

INPUT: select Service, Sub Service, Service Provider Name and Type of AMC

PROCESS: It will enter this details in database

OUTPUT: “Your AMC Booked” message display on screen

**6.Give Feedback about services**

INPUT: Enter User Email-Id, Service Provider name and Feedback

PROCESS: This Feedback store in database with date

OUTPUT: “Thanks for giving Feedback” message on the screen

**7.Provide Tips for Maintenance**

INPUT: Select Object or machine

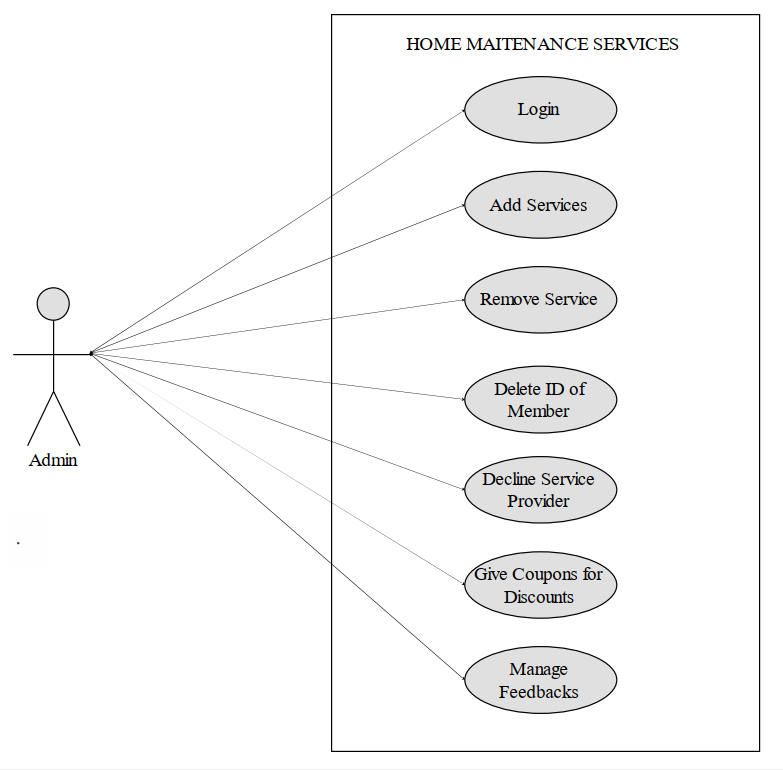
PROCESS: Fetch tips

OUTPUT: Display solutions

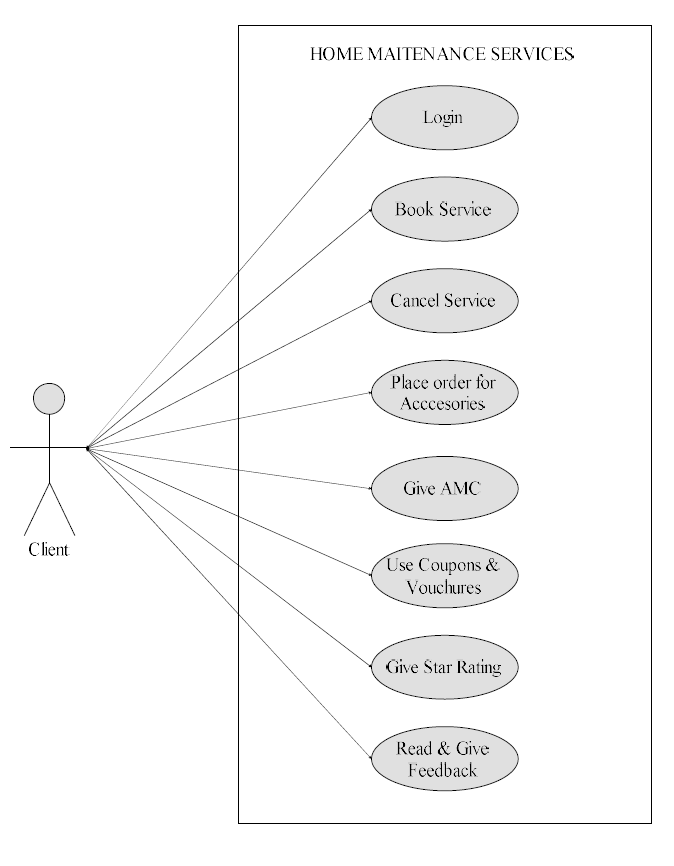
**3. SYSTEM ANALYSIS MODELING-USER BASED**

**3.1 USER-BASED MODELING**

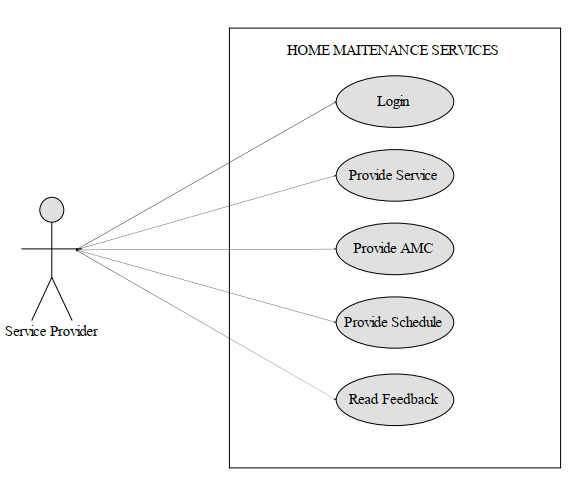
**3.1.1 USE CASE FOR ADMIN**



**3.1.2 USE CASE FOR CLIENT**

****

**3.1.3 USE CASE FOR SERVICE PROVIDER**



**4. SYSTEM ANALYSIS AND DESIGN – DATA BASED**

**4.1 DATA MODELING**

**4.1.1 DATA DICTIONARY**

**1. admin**

**DROP TABLE IF EXISTS `admin`;**

**CREATE TABLE IF NOT EXISTS `admin` (**

**`rowid` int(11) NOT NULL AUTO\_INCREMENT,**

**`Name` varchar(20) NOT NULL,**

**`Contact` int(11) NOT NULL,**

**`Address` varchar(100) NOT NULL,**

**`A\_Email\_Id` varchar(40) NOT NULL,**

**`A\_Password` varchar(10) NOT NULL,**

**`City` varchar(15) NOT NULL,**

**PRIMARY KEY (`rowid`)**

**);**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Datatype** | **Size** | **Constraint** |
| rowid | int | 11 | PRIMARY KEY |
| Name | VARCHAR | 10 | NOT NULL |
| Contact | int | 11 | NOT NULL |
| Address | varchar | 100 | NOT NULL |
| A\_Email\_Id | varchar | 40 | NOT NULL |
| A\_Password | varchar | 10 | NOT NULL |
| City | varchar | 15 | NOT NULL |

**2. client\_details**

**DROP TABLE IF EXISTS `client\_details`;**

**CREATE TABLE IF NOT EXISTS `client\_details` (**

**`C\_Name` varchar(40) NOT NULL,**

**`C\_Email\_Id` varchar(40) NOT NULL,**

**`C\_Contact` int(11) NOT NULL,**

**`C\_City` varchar(20) NOT NULL,**

**`C\_Address` varchar(100) NOT NULL,**

**`C\_Password` varchar(10) NOT NULL,**

**`C\_Total\_Service\_Number` int NOT NULL,**

**`C\_Otp` int(11) NOT NULL,**

**PRIMARY KEY (`C\_Email\_Id`)**

**);**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Datatype** | **Size** | **Constraint** |
| C\_Name | varchar | 40 | NOT NULL |
| C\_Email\_Id | varchar | 40 | Primary Key |
| C\_Contact | int | 11 | NOT NULL |
| C\_City | varchar | 20 | NOT NULL |
| C\_Address | varchar | 100 | NOT NULL |
| C\_Password | varchar | 10 | NOT NULL |
| C\_Total\_Service | int | 11 | NOT NULL |
| C\_Otp | int | 11 | NOT NULL |

**3. client\_service\_details**

**DROP TABLE IF EXISTS `client\_service\_details`;**

**CREATE TABLE IF NOT EXISTS `client\_service\_details` (**

**`Record\_No` int(11) NOT NULL AUTO\_INCREMENT,**

**`Client\_Email\_Id` varchar(40) NOT NULL,**

**`Service\_Id` varchar(10) NOT NULL,**

**`Sub\_Service\_Id` varchar(10) NOT NULL,**

**`S\_P\_Id` varchar(10) NOT NULL,**

**`C\_Address` varchar(100) NOT NULL,**

**`C\_Date` date NOT NULL,**

**`P\_Date` date NOT NULL,**

**`Provided` tinyint(1) NOT NULL,**

**`Payment` tinyint(1) NOT NULL,**

**PRIMARY KEY (`Record\_No`),**

**KEY `FK1` (`Service\_Id`) USING BTREE,**

**KEY `FK2` (`S\_P\_Id`),**

**KEY `FK3` (`Sub\_Service\_Id`));**

**ALTER TABLE `client\_service\_details`**

**ADD CONSTRAINT `FK3` FOREIGN KEY (`Sub\_Service\_Id`) REFERENCES `sub\_service` (`S\_Sub\_Id`) ON DELETE CASCADE ON UPDATE CASCADE;**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Datatype** | **Size** | **Constraint** |
| Record\_No | int | 40 | PRIMARY KEY |
| Client\_Email\_Id | varchar | 40 | NOT NULL |
| Service\_Id | varchar | 10 | FOREIGN KEY (FK1), NOT NULL |
| Sub\_Service\_Id | varchar | 10 | FOREIGN KEY (FK2), NOT NULL |
| C\_Address | varchar | 100 | NOT NULL |
| C\_Date | date |  | NOT NULL |
| P\_Date | date |  | NOT NULL |
| Provided | tinyint | 1 | NOT NULL |
| Payment | tinyint | 1 | NOT NULL |

**4. contract\_details**

**DROP TABLE IF EXISTS `contract\_details`;**

**CREATE TABLE IF NOT EXISTS `contract\_details` (**

**`C\_Id` int(11) NOT NULL AUTO\_INCREMENT,**

**`C\_Email\_Id` varchar(40) NOT NULL,**

**`Service\_Id` varchar(10) NOT NULL,**

**`S\_P\_Id` varchar(10) NOT NULL,**

**`S\_C\_Plan` varchar(20) NOT NULL,**

**`C\_Start\_Date` date NOT NULL,**

**`C\_End\_Date` date NOT NULL,**

**`Total\_Cost` int(5) NOT NULL,**

**`Payment` tinyint(1) NOT NULL,**

**PRIMARY KEY (`C\_Id`),**

**KEY `FK6` (`C\_Email\_Id`),**

**KEY `FK7` (`Service\_Id`),**

**KEY `FK8` (`S\_P\_Id`));**

**ALTER TABLE `contract\_details`**

**ADD CONSTRAINT `FK6` FOREIGN KEY (`C\_Email\_Id`) REFERENCES `client\_details` (`C\_Email\_Id`) ON DELETE CASCADE ON UPDATE CASCADE,**

**ADD CONSTRAINT `FK7` FOREIGN KEY (`Service\_Id`) REFERENCES `service` (`Service\_Id`) ON DELETE CASCADE ON UPDATE CASCADE,**

**ADD CONSTRAINT `FK8` FOREIGN KEY (`S\_P\_Id`) REFERENCES `service\_provider\_details` (`S\_P\_Id`) ON DELETE CASCADE ON UPDATE CASCADE;**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Datatype** | **Size** | **Constraint** |
| C\_Id | int | 11 | PRIMARY KEY |
| C\_Email\_Id | varchar | 40 | FOREIGN KEY (FK6), NOT NULL |
| Service\_Id | varchar | 10 | FOREIGN KEY (FK7), NOT NULL |
| S\_P\_Id | varchar | 10 | FOREIGN KEY (FK8), NOT NULL |
| S\_C\_Plan | varchar | 20 | NOT NULL |
| C\_Start\_Date | date |  | NOT NULL |
| C\_End\_Date | date |  | NOT NULL |
| Total\_Cost | int | 5 | NOT NULL |
| Payment | tinyint | 1 | NOT NULL |

**5. contract\_record**

**DROP TABLE IF EXISTS `contract\_record`;**

**CREATE TABLE IF NOT EXISTS `contract\_record` (**

**`Id` int(11) NOT NULL AUTO\_INCREMENT,**

**`C\_Id` int(11) NOT NULL,**

**`C\_Date` date NOT NULL,**

**`P\_Date` date NOT NULL,**

**`Provided` tinyint(1) NOT NULL,**

**PRIMARY KEY (`Id`),**

**KEY `FK9` (`C\_Id`)**

**);**

***--***

***-- Constraints for table `contract\_record`***

***--***

**ALTER TABLE `contract\_record`**

**ADD CONSTRAINT `FK9` FOREIGN KEY (`C\_Id`) REFERENCES `contract\_details` (`C\_Id`) ON DELETE CASCADE ON UPDATE CASCADE;**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Datatype** | **Size** | **Constraint** |
| Id | int | 11 | PRIMARY KEY |
| C\_Id | int | 11 | FOREIGN KEY (FK9), NOT NULL |
| C\_Date | date |  | NOT NULL |
| P\_Date | date |  | NOT NULL |
| Provided | tinyint | 1 | NOT NULL |

**6. feedback\_details**

**DROP TABLE IF EXISTS `feedback\_details`;**

**CREATE TABLE IF NOT EXISTS `feedback\_details` (**

**`F\_Id` int(11) NOT NULL AUTO\_INCREMENT,**

**`C\_Email\_Id` varchar(40) NOT NULL,**

**`C\_Feedback` varchar(500) NOT NULL,**

**`C\_F\_Date` date NOT NULL,**

**`Service\_Id` varchar(10) NOT NULL,**

**`S\_P\_Id` varchar(10) NOT NULL,**

**PRIMARY KEY (`F\_Id`),**

**KEY `Foreign key` (`Service\_Id`),**

**KEY `FK4` (`S\_P\_Id`),**

**KEY `FK5` (`C\_Email\_Id`)**

**);**

***--***

***-- Constraints for table `feedback\_details`***

***--***

**ALTER TABLE `feedback\_details`**

**ADD CONSTRAINT `FK4` FOREIGN KEY (`S\_P\_Id`) REFERENCES `service\_provider\_details` (`S\_P\_Id`) ON DELETE CASCADE ON UPDATE CASCADE,**

**ADD CONSTRAINT `FK5` FOREIGN KEY (`C\_Email\_Id`) REFERENCES `client\_details` (`C\_Email\_Id`) ON DELETE CASCADE ON UPDATE CASCADE,**

**ADD CONSTRAINT `Foreign key` FOREIGN KEY (`Service\_Id`) REFERENCES `service` (`Service\_Id`) ON DELETE CASCADE ON UPDATE CASCADE;**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Datatype** | **Size** | **Constraint** |
| F\_Id | int | 11 | PRIMARY KEY |
| C\_Email\_Id | varchar | 40 | FOREIGN KEY (FK5), NOT NULL |
| C\_Feedback | varchar | 500 | NOT NULL |
| C\_F\_Date | date | - | NOT NULL |
| Service\_Id | varchar | 10 | FOREIGN KEY (Foreing Key), NOT NULL |
| S\_P\_Id | VARCHAR | 10 | FOREIGN KEY (FK4), NOT NULL |

**7. service**

**DROP TABLE IF EXISTS `service`;**

**CREATE TABLE IF NOT EXISTS `service` (**

**`Service\_Id` varchar(10) NOT NULL,**

**`Service\_Name` varchar(40) NOT NULL,**

**`C\_Price` int(5) NOT NULL,**

**PRIMARY KEY (`Service\_Id`)**

**);**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Datatype** | **Size** | **Constraint** |
| Service\_Id | varchar | 10 | PRIMARY KEY |
| Service\_Name | varchar | 40 | NOT NULL |
| C\_Price | int | 5 | NOT NULL |

**8. service\_provider\_detail**

**DROP TABLE IF EXISTS `service\_provider\_details`;**

**CREATE TABLE IF NOT EXISTS `service\_provider\_details` (**

**`S\_P\_Id` varchar(10) NOT NULL,**

**`S\_P\_Name` varchar(40) NOT NULL,**

**`S\_P\_Experience\_Year` int(2) NOT NULL,**

**`S\_P\_Email\_Id` varchar(40) NOT NULL,**

**`S\_P\_Contact` text NOT NULL,**

**`S\_P\_City` varchar(20) NOT NULL,**

**`S\_P\_Address` varchar(100) NOT NULL,**

**`S\_P\_Password` varchar(10) NOT NULL,**

**PRIMARY KEY (`S\_P\_Id`)**

**);**

***--***

***-- Constraints for table `service\_provider\_services`***

***--***

**ALTER TABLE `service\_provider\_services`**

**ADD CONSTRAINT `FK11` FOREIGN KEY (`Service\_Id`) REFERENCES `service` (`Service\_Id`) ON DELETE CASCADE ON UPDATE CASCADE,**

**ADD CONSTRAINT `FK12` FOREIGN KEY (`S\_P\_Id`) REFERENCES `service\_provider\_details` (`S\_P\_Id`) ON DELETE CASCADE ON UPDATE CASCADE;**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Datatype** | **Size** | **Constraint** |
| S\_P\_Id | varchar | 10 | PRIMARY KEY |
| S\_P\_Name | varchar | 40 | NOT NULL |
| S\_P\_Experience\_Year | int | 2 | NOT NULL |
| S\_P\_Email\_Id | varchar | 40 | NOT NULL |
| S\_P\_Contact | int | 11 | NOT NULL |
| S\_P\_City | varchar | 20 | NOT NULL |
| S\_P\_Address | varchar | 100 | NOT NULL |
| S\_P\_Password | varchar | 10 | NOT NULL |

**9. service\_provider\_services**

**DROP TABLE IF EXISTS `service\_provider\_services`;**

**CREATE TABLE IF NOT EXISTS `service\_provider\_services` (**

**`S\_Extra` int(11) NOT NULL AUTO\_INCREMENT,**

**`Service\_Id` varchar(10) NOT NULL,**

**`S\_P\_Id` varchar(10) NOT NULL,**

**PRIMARY KEY (`S\_Extra`),**

**KEY `FK11` (`Service\_Id`),**

**KEY `FK12` (`S\_P\_Id`)**

**);**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Datatype** | **Size** | **Constraint** |
| S\_Extra | int | 11 | PRIMARY KEY |
| Service\_Id | varchar | 10 | FOREIGN KEY (FK11), NOT NULL |
| S\_P\_Id | varchar | 10 | FOREIGN KEY (FK12), NOT NULL |

**10. sub\_service**

**DROP TABLE IF EXISTS `sub\_service`;**

**CREATE TABLE IF NOT EXISTS `sub\_service` (**

**`Service\_Id` varchar(10) NOT NULL,**

**`S\_Sub\_Id` varchar(10) NOT NULL,**

**`S\_Sub\_Name` varchar(40) NOT NULL,**

**`S\_Sub\_Charge` int(5) NOT NULL,**

**PRIMARY KEY (`S\_Sub\_Id`),**

**KEY `FK1` (`Service\_Id`)**

**);**

***--***

***-- Constraints for table `sub\_service`***

***--***

**ALTER TABLE `sub\_service`**

**ADD CONSTRAINT `FK1` FOREIGN KEY (`Service\_Id`) REFERENCES `service` (`Service\_Id`) ON DELETE CASCADE ON UPDATE CASCADE;**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Datatype** | **Size** | **Constraint** |
| Service\_Id | varchar | 10 | FOREIGN KEY (FK1), NOT NULL |
| S\_Sub\_Id | varchar | 10 | PRIMARY KEY |
| S\_Sub\_Name | varchar | 40 | NOT NULL |
| S\_Sub\_Charge | int | 5 | NOT NULL |

**Procedures**

***--***

***-- Procedures***

***--***

**DROP PROCEDURE IF EXISTS `FeedbackInsert`$$**

**CREATE DEFINER=`root`@`localhost` PROCEDURE `FeedbackInsert` (IN `email` VARCHAR(40), IN `feedback` VARCHAR(500), IN `date` DATE, IN `S\_id` VARCHAR(10), IN `S\_P\_id` VARCHAR(10)) BEGIN**

**insert into feedback\_details values(NULL,email,feedback,date,S\_id,'S\_P\_id');**

**END$$**

**DROP PROCEDURE IF EXISTS `GetUserByEmail`$$**

**CREATE DEFINER=`root`@`localhost` PROCEDURE `GetUserByEmail` (IN `eid` VARCHAR(40)) BEGIN**

**select C\_Name from client\_details where C\_Email\_Id=eid;**

**END$$**

**DROP PROCEDURE IF EXISTS `service`$$**

**CREATE DEFINER=`root`@`localhost` PROCEDURE `service` () BEGIN**

**select \* from service;**

**END$$**

**DROP PROCEDURE IF EXISTS `servicename`$$**

**CREATE DEFINER=`root`@`localhost` PROCEDURE `servicename` (IN `Name` VARCHAR(40)) BEGIN**

**SELECT Service\_Id from service where Service\_Name=Name;**

**END$$**

**DROP PROCEDURE IF EXISTS `service\_provider\_services\_insert`$$**

**CREATE DEFINER=`root`@`localhost` PROCEDURE `service\_provider\_services\_insert` (IN `sid` VARCHAR(10), IN `spid` VARCHAR(10)) BEGIN**

**insert into service\_provider\_services values(null,sid,spid);**

**END$$**

**DROP PROCEDURE IF EXISTS `subservice`$$**

**CREATE DEFINER=`root`@`localhost` PROCEDURE `subservice` (IN `eid` VARCHAR(10)) BEGIN**

**select \* from sub\_service where Service\_Id=eid;**

**END$$**

**DROP PROCEDURE IF EXISTS `sub\_service`$$**

**CREATE DEFINER=`root`@`localhost` PROCEDURE `sub\_service` (IN `s` VARCHAR(10)) BEGIN**

**SELECT \* FROM sub\_service s WHERE s.S\_Sub\_Id IN (SELECT S\_Sub\_Id FROM `sub\_service` WHERE Service\_Id=s);**

**END$$**

**DROP PROCEDURE IF EXISTS `total\_number\_of\_service\_provider`$$**

**CREATE DEFINER=`root`@`localhost` PROCEDURE `total\_number\_of\_service\_provider` () BEGIN**

**SELECT COUNT(DISTINCT S\_P\_Id) FROM service\_provider\_details;**

**END$$**

**DROP PROCEDURE IF EXISTS `update\_address`$$**

**CREATE DEFINER=`root`@`localhost` PROCEDURE `update\_address` (IN `Address` VARCHAR(100), IN `e` VARCHAR(40)) BEGIN**

**update client\_details set client\_details.C\_Address=Address where client\_details.C\_Email\_Id=e;**

**END$$**

**DROP PROCEDURE IF EXISTS `update\_otp`$$**

**CREATE DEFINER=`root`@`localhost` PROCEDURE `update\_otp` (IN `str` INT(11), IN `e` VARCHAR(40)) BEGIN**

**update client\_details set C\_Otp=str where C\_Email\_Id=e;**

**END$$**

**DROP PROCEDURE IF EXISTS `update\_service\_provider\_name`$$**

**CREATE DEFINER=`root`@`localhost` PROCEDURE `update\_service\_provider\_name` (IN `Name` VARCHAR(40), IN `spid` VARCHAR(10)) BEGIN**

**update service\_provider\_details set S\_P\_Name=Name where S\_P\_Id=spid;**

**END$$**

**Functions**

**DROP FUNCTION IF EXISTS `Expericence\_of\_Service\_Provider`$$**

**CREATE DEFINER=`root`@`localhost` FUNCTION `Expericence\_of\_Service\_Provider` (`Name` VARCHAR(40)) RETURNS INT(2) BEGIN**

**DECLARE Y VARCHAR(40) DEFAULT"";**

**select S\_P\_Experience\_Year INTO Y from service\_provider\_details where S\_P\_Name=Name;**

**RETURN Y;**

**END$$**

**DROP FUNCTION IF EXISTS `GetOtp`$$**

**CREATE DEFINER=`root`@`localhost` FUNCTION `GetOtp` (`e` VARCHAR(40)) RETURNS INT(11) UNSIGNED BEGIN**

**DECLARE Otp VARCHAR(40) DEFAULT"";**

**select C\_Otp into Otp from client\_details where C\_Email\_Id=e;**

**RETURN Otp;**

**END$$**

**DROP FUNCTION IF EXISTS `GetPassword`$$**

**CREATE DEFINER=`root`@`localhost` FUNCTION `GetPassword` (`eid` VARCHAR(40)) RETURNS VARCHAR(10) CHARSET latin1 BEGIN**

**DECLARE Pass VARCHAR(10) DEFAULT"";**

**select C\_Password into Pass from client\_details where C\_Email\_Id=eid;**

**RETURN Pass;**

**END$$**

**DROP FUNCTION IF EXISTS `GetUserByEmailFunction`$$**

**CREATE DEFINER=`root`@`localhost` FUNCTION `GetUserByEmailFunction` (`eid` VARCHAR(40)) RETURNS VARCHAR(40) CHARSET latin1 BEGIN**

**DECLARE Name VARCHAR(40) DEFAULT"";**

**select C\_Name INTO Name from client\_details where C\_Email\_Id=eid;**

**RETURN Name;**

**END$$**

**DROP FUNCTION IF EXISTS `Get\_Service\_Price`$$**

**CREATE DEFINER=`root`@`localhost` FUNCTION `Get\_Service\_Price` (`SID` VARCHAR(10)) RETURNS INT(5) BEGIN**

**DECLARE Price VARCHAR(40) DEFAULT"";**

**select C\_Price INTO Price from service where Service\_Id=SID;**

**RETURN Price;**

**END$$**

**DELIMITER ;**

**Triggers**

***--***

***-- Triggers `admin`***

***--***

**DROP TRIGGER IF EXISTS `admin\_contact`;**

**DELIMITER $$**

**CREATE TRIGGER `admin\_contact` BEFORE INSERT ON `admin` FOR EACH ROW BEGIN**

**IF (LENGTH(NEW.Contact) != 10) THEN**

**SIGNAL SQLSTATE '02000' SET MESSAGE\_TEXT = 'Warning: Contact Number must be 10 digits.';**

**END IF;**

**END**

**$$**

**DELIMITER ;**

**DROP TRIGGER IF EXISTS `admin\_mail`;**

**DELIMITER $$**

**CREATE TRIGGER `admin\_mail` BEFORE INSERT ON `admin` FOR EACH ROW BEGIN**

**if (EXISTS( SELECT \* from admin where admin.A\_Email\_Id=new.A\_Email\_Id)) THEN**

**signal sqlstate '47000' set message\_text ='These email is already in use.';**

**END IF;**

**END**

**$$**

**DELIMITER ;**

**DROP TRIGGER IF EXISTS `admin\_name`;**

**DELIMITER $$**

**CREATE TRIGGER `admin\_name` BEFORE INSERT ON `admin` FOR EACH ROW BEGIN**

**if (EXISTS( SELECT \* from admin where admin.Name=new.Name)) THEN**

**signal sqlstate '47000' set message\_text ='These username is already exists';**

**END IF;**

**END**

**$$**

**DELIMITER ;**

**DROP TRIGGER IF EXISTS `admin\_password`;**

**DELIMITER $$**

**CREATE TRIGGER `admin\_password` BEFORE INSERT ON `admin` FOR EACH ROW BEGIN**

**IF (LENGTH(NEW.A\_Password) <= 8) THEN**

**SIGNAL SQLSTATE '02000' SET MESSAGE\_TEXT = 'Warning: Password must be greater than or equal to 8 digits.';**

**END IF;**

**END**

**$$**

**DELIMITER ;**

**DROP TRIGGER IF EXISTS `restrictiond`;**

**DELIMITER $$**

**CREATE TRIGGER `restrictiond` BEFORE DELETE ON `admin` FOR EACH ROW BEGIN**

**if (current\_time() > CAST('08:30:00' AS time) OR current\_time() < CAST('18:30:00' AS time)) then**

**SIGNAL SQLSTATE '02000' SET MESSAGE\_TEXT = 'Error: Not allowed to delete data at this time....';**

**end if;**

**END**

**$$**

**DELIMITER ;**

**DROP TRIGGER IF EXISTS `restrictioni`;**

**DELIMITER $$**

**CREATE TRIGGER `restrictioni` BEFORE INSERT ON `admin` FOR EACH ROW BEGIN**

**if (current\_time() > CAST('08:30:00' AS time) OR current\_time() < CAST('18:30:00' AS time)) then**

**SIGNAL SQLSTATE '02000' SET MESSAGE\_TEXT = 'Error: Not allowed to insert data at this time....';**

**end if;**

**END**

**$$**

**DELIMITER ;**

**DROP TRIGGER IF EXISTS `restrictionu`;**

**DELIMITER $$**

**CREATE TRIGGER `restrictionu` BEFORE UPDATE ON `admin` FOR EACH ROW BEGIN**

**if (current\_time() > CAST('08:30:00' AS time) OR current\_time() < CAST('18:30:00' AS time)) then**

**SIGNAL SQLSTATE '02000' SET MESSAGE\_TEXT = 'Error: Not allowed to update data at this time....';**

**end if;**

**END**

**$$**

**DELIMITER ;**

***--***

***-- Triggers `client\_details`***

***--***

**DROP TRIGGER IF EXISTS `Name`;**

**DELIMITER $$**

**CREATE TRIGGER `Name` BEFORE INSERT ON `client\_details` FOR EACH ROW BEGIN**

**if (EXISTS( SELECT \* from client\_details where client\_details.C\_Name=new.C\_Name)) THEN**

**signal sqlstate '47000' set message\_text ='These username is already exists';**

**END IF;**

**END**

**$$**

**DELIMITER ;**

**DROP TRIGGER IF EXISTS `password`;**

**DELIMITER $$**

**CREATE TRIGGER `password` BEFORE INSERT ON `client\_details` FOR EACH ROW BEGIN**

**IF (LENGTH(NEW.C\_Password) <= 8) THEN**

**SIGNAL SQLSTATE '02000' SET MESSAGE\_TEXT = 'Warning: Password must be greater than or equal to 8 digits.';**

**END IF;**

**END**

**$$**

**DELIMITER ;**

**DROP TRIGGER IF EXISTS `phone`;**

**DELIMITER $$**

**CREATE TRIGGER `phone` BEFORE INSERT ON `client\_details` FOR EACH ROW BEGIN**

**IF (LENGTH(NEW.C\_Contact) != 10) THEN**

**SIGNAL SQLSTATE '02000' SET MESSAGE\_TEXT = 'Warning: Contact Number must be 10 digits.';**

**END IF;**

**END**

**$$**

**DELIMITER ;**

***--***

***-- Triggers `feedback\_details`***

***--***

**DROP TRIGGER IF EXISTS `feedback\_added`;**

**DELIMITER $$**

**CREATE TRIGGER `feedback\_added` AFTER INSERT ON `feedback\_details` FOR EACH ROW BEGIN**

**SIGNAL SQLSTATE '02000' SET MESSAGE\_TEXT = 'Thanks for giving Feedback';**

**END**

**$$**

**DELIMITER ;**

***--***

***-- Triggers `service\_provider\_details`***

***--***

**DROP TRIGGER IF EXISTS `S\_P\_Contact`;**

**DELIMITER $$**

**CREATE TRIGGER `S\_P\_Contact` BEFORE INSERT ON `service\_provider\_details` FOR EACH ROW BEGIN**

**IF (LENGTH(NEW.S\_P\_Contact) != 10) THEN**

**SIGNAL SQLSTATE '02000' SET MESSAGE\_TEXT = 'Warning: Contact Number must be 10 digits.';**

**END IF;**

**END**

**$$**

**DELIMITER ;**

**DROP TRIGGER IF EXISTS `S\_P\_Email`;**

**DELIMITER $$**

**CREATE TRIGGER `S\_P\_Email` BEFORE INSERT ON `service\_provider\_details` FOR EACH ROW BEGIN**

**if (EXISTS( SELECT \* from admin where service\_provider\_details.S\_P\_Email\_Id=new.S\_P\_Email\_Id)) THEN**

**signal sqlstate '47000' set message\_text ='This email is already in use.';**

**END IF;**

**END**

**$$**

**DELIMITER ;**

**DROP TRIGGER IF EXISTS `S\_P\_Password`;**

**DELIMITER $$**

**CREATE TRIGGER `S\_P\_Password` BEFORE INSERT ON `service\_provider\_details` FOR EACH ROW BEGIN**

**IF (LENGTH(NEW.S\_P\_Password) <= 8) THEN**

**SIGNAL SQLSTATE '02000' SET MESSAGE\_TEXT = 'Warning: Password must be greater than or equal to 8 digits.';**

**END IF;**

**END**

**$$**

**DELIMITER ;**

***--***

***-- Triggers `service\_provider\_services`***

***--***

**DROP TRIGGER IF EXISTS `Service\_Added\_Successfully`;**

**DELIMITER $$**

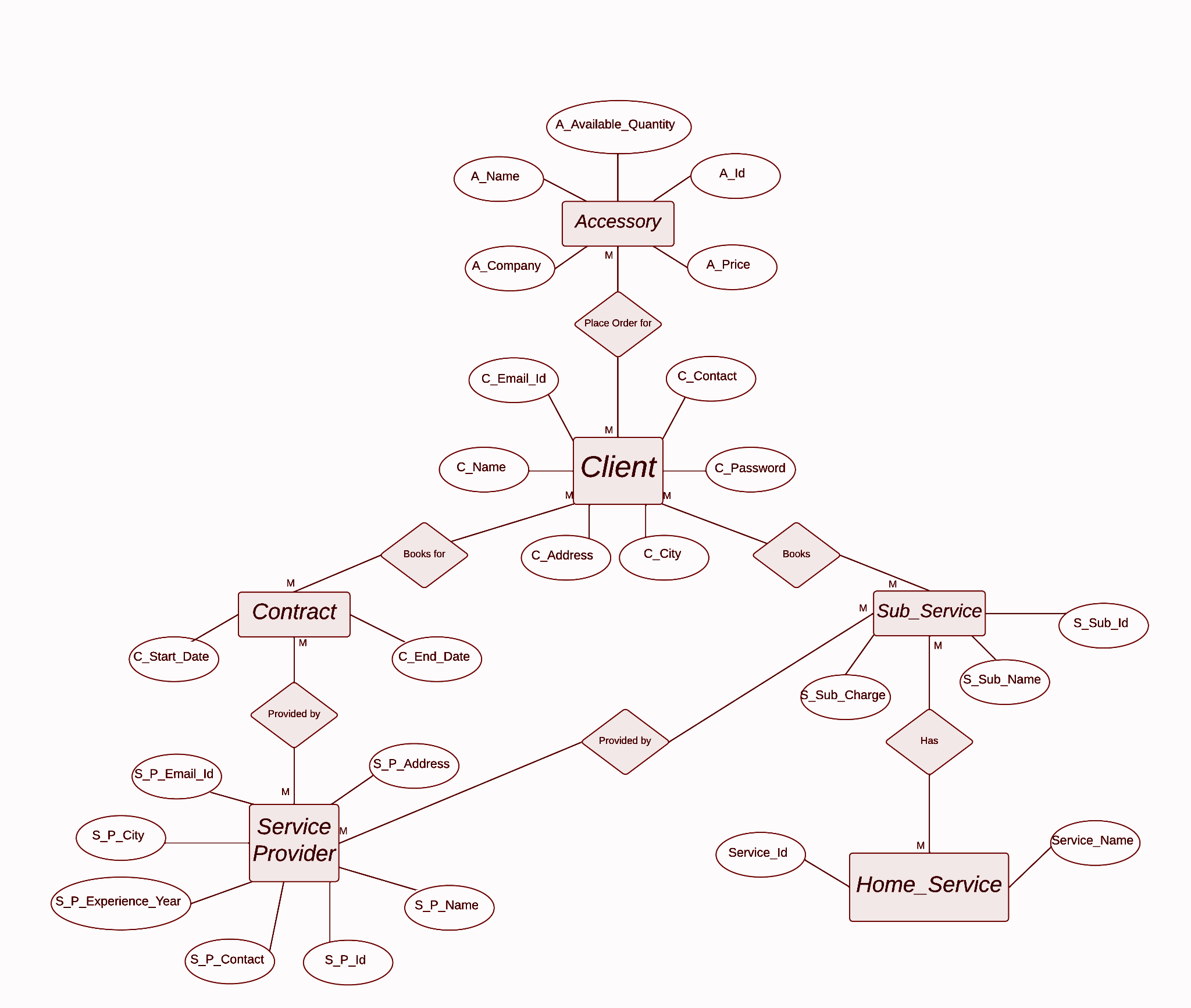
**CREATE TRIGGER `Service\_Added\_Successfully` AFTER INSERT ON `service\_provider\_services` FOR EACH ROW BEGIN**

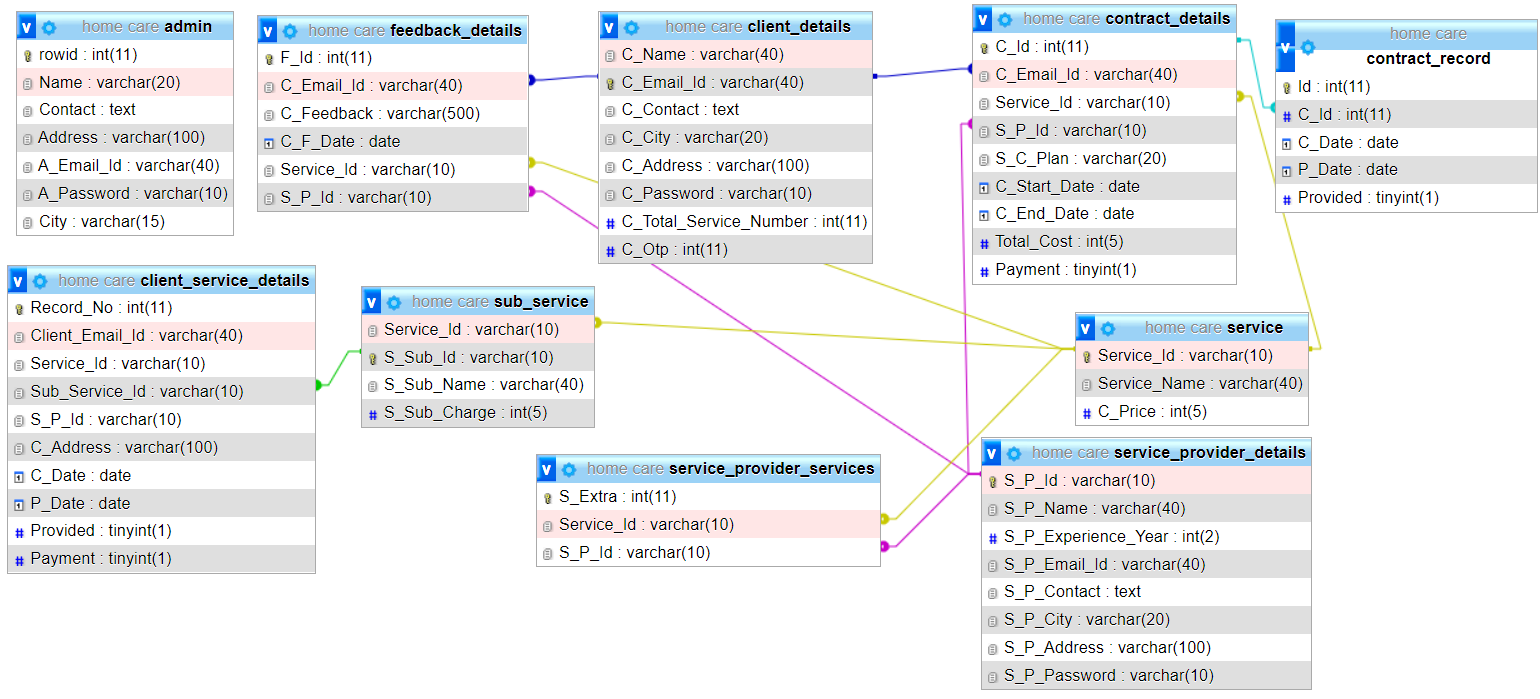
**SIGNAL SQLSTATE '02000' SET MESSAGE\_TEXT = 'Service Added Successfully';**

**END**

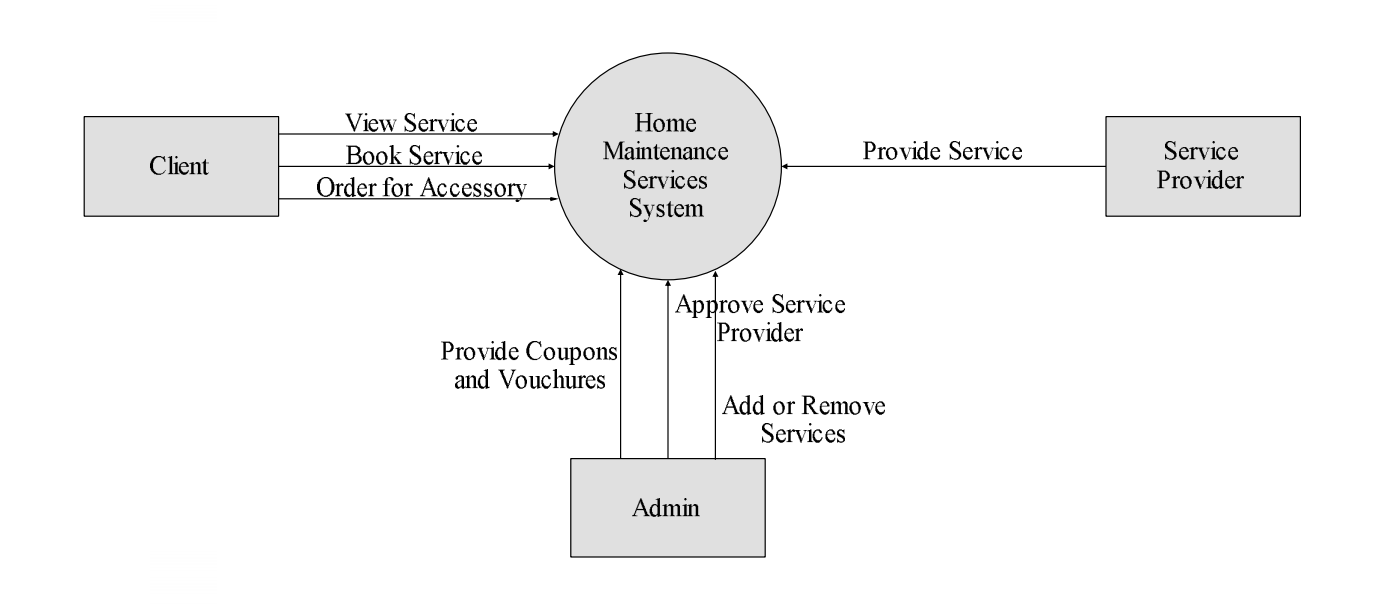
**$$**

**DELIMITER ;**

**4.1.2 E-R DIAGRAM**



**4.2 BEHAVIORAL MODELING**

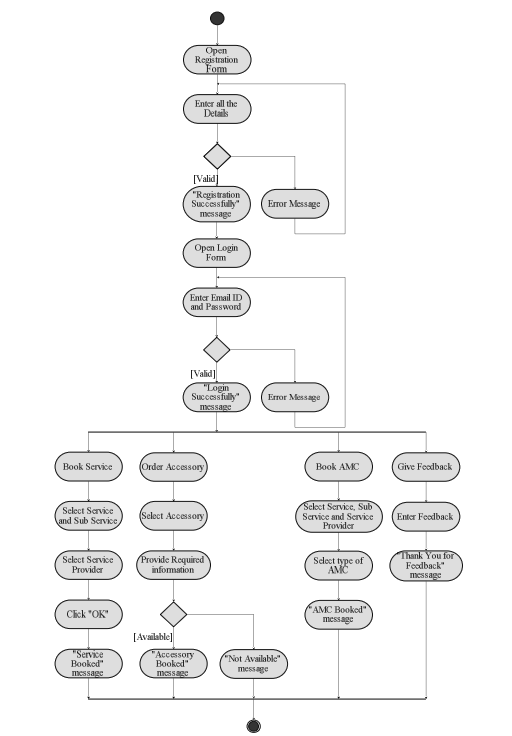
****

**5. SYSTEM DESIGN – UML**

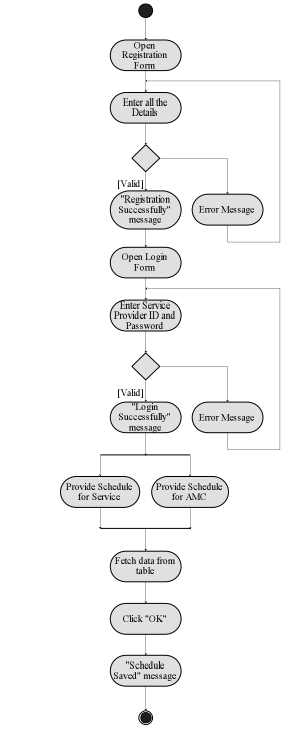
**5.1 ACTIVITY DIAGRAMS**

**ADMIN**



**CLIENT**

**SERVICE PROVIDER**

****

**6. SYSTEM INTERFACE DESIGN**

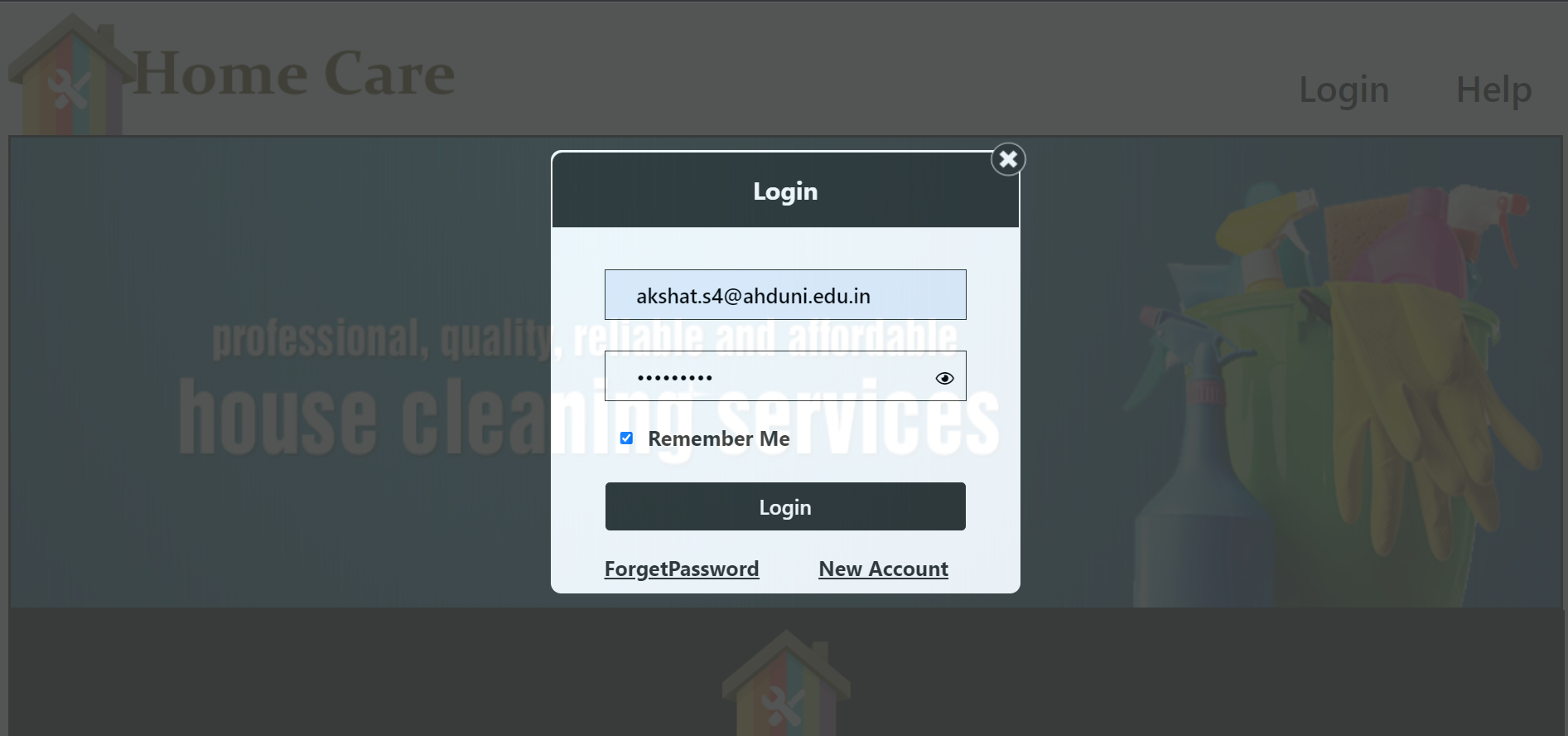
**6.1 GRAPHICAL USER INTERFACE**

**HOME PAGE**

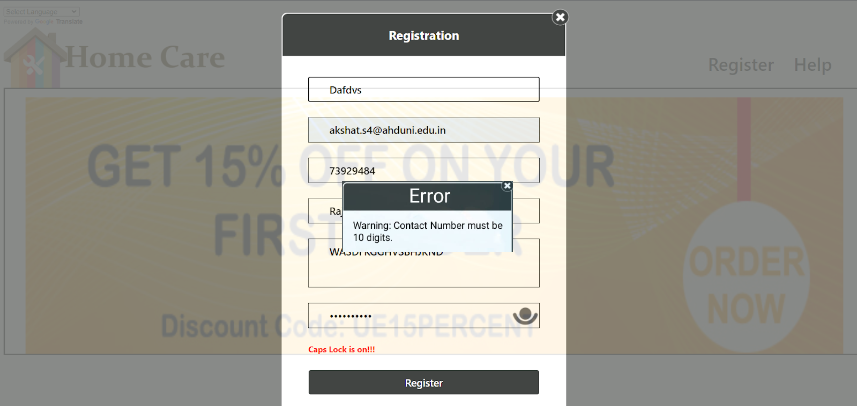
**CLIENT REGISTER**

****

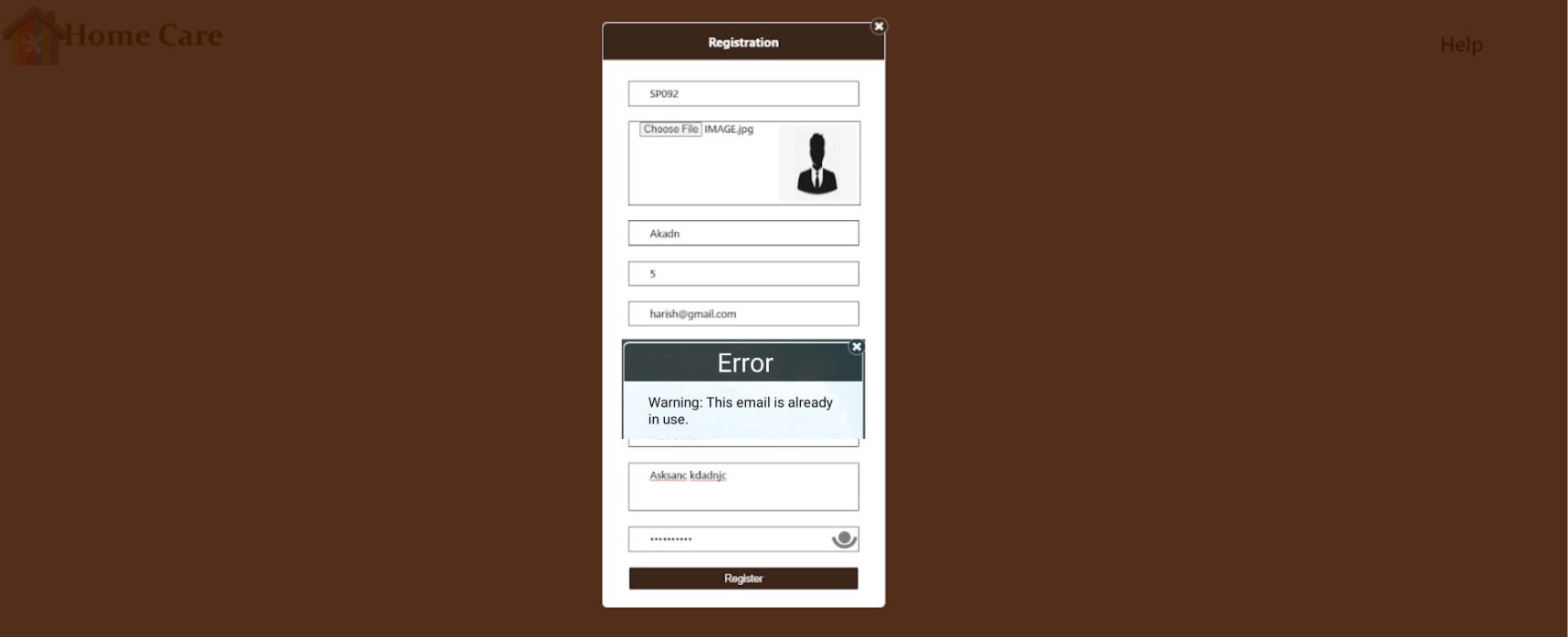
**CLIENT LOGIN**

****

**FIRING TRIGGER FOR CONTACT NO.**

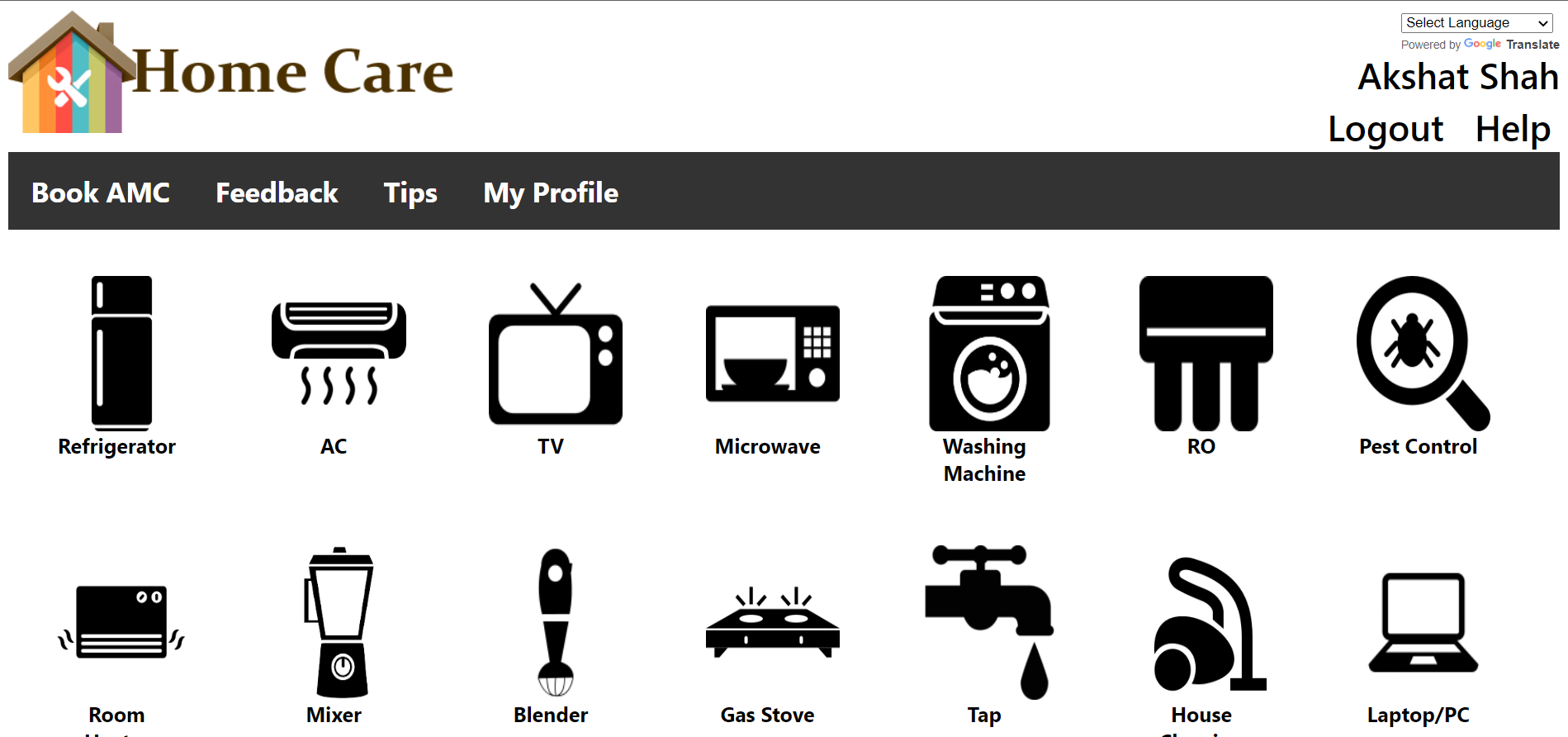
****

**FIRING TRIGGER FOR SERVICE PROVIDER REGISTRATION**

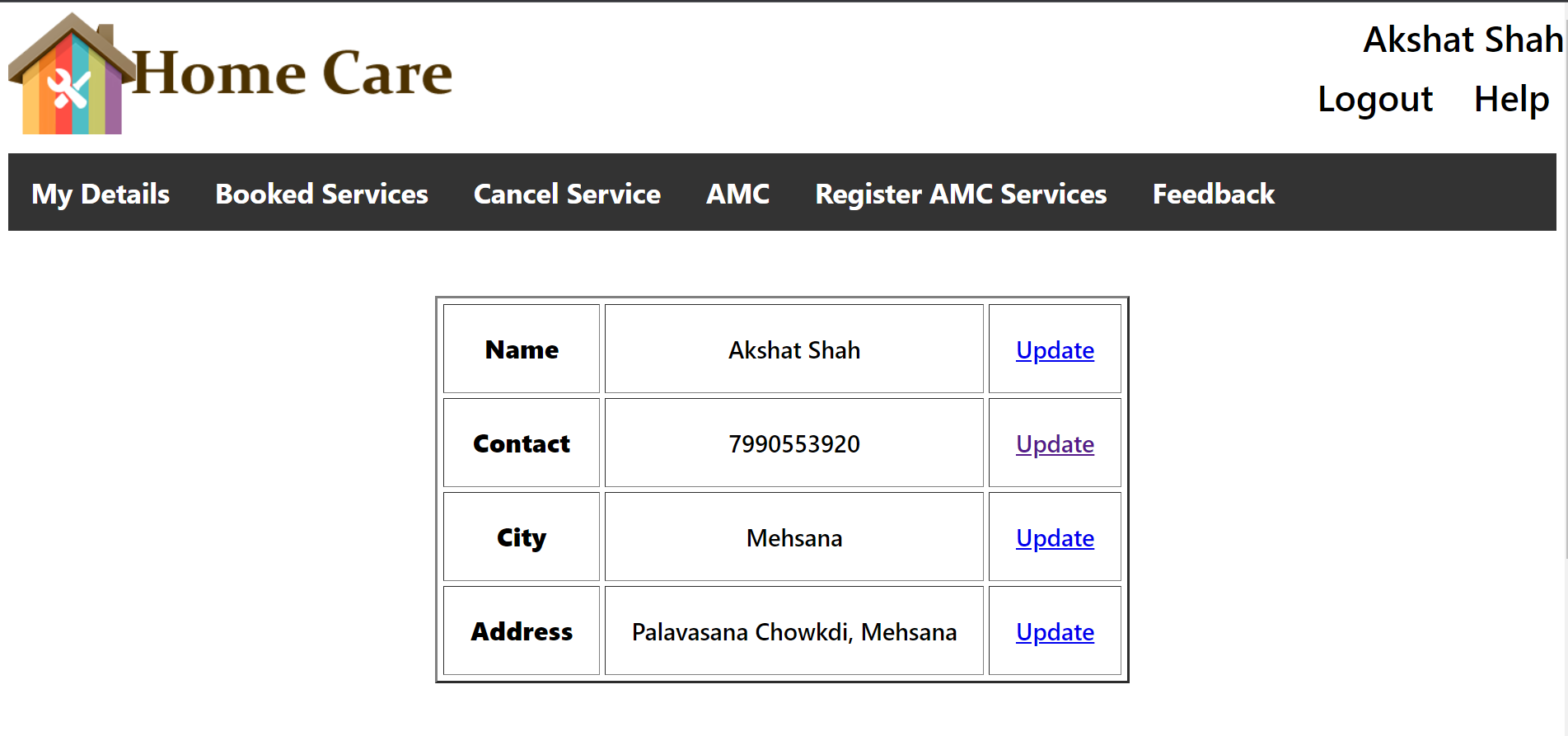
****

**SERVICES**

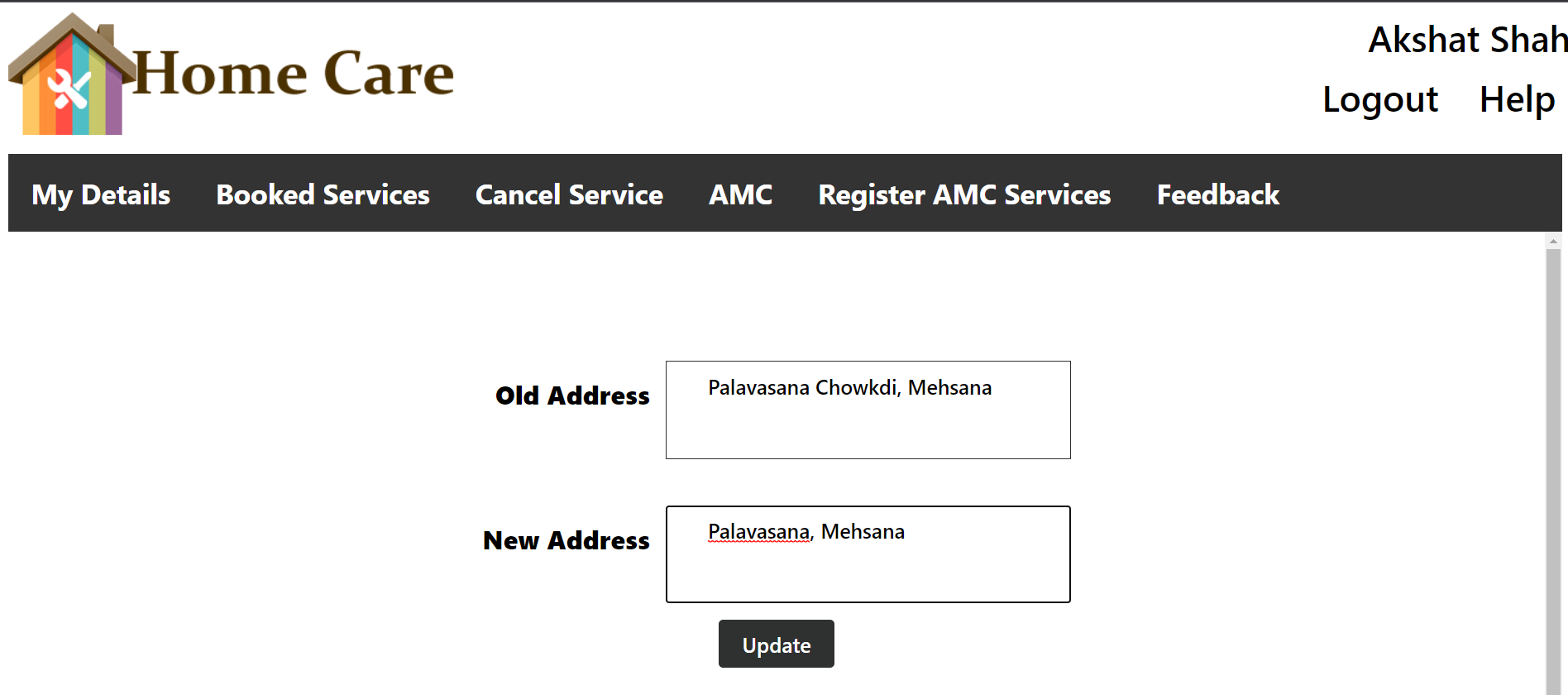
**(Function used in the Name displayed)**

****

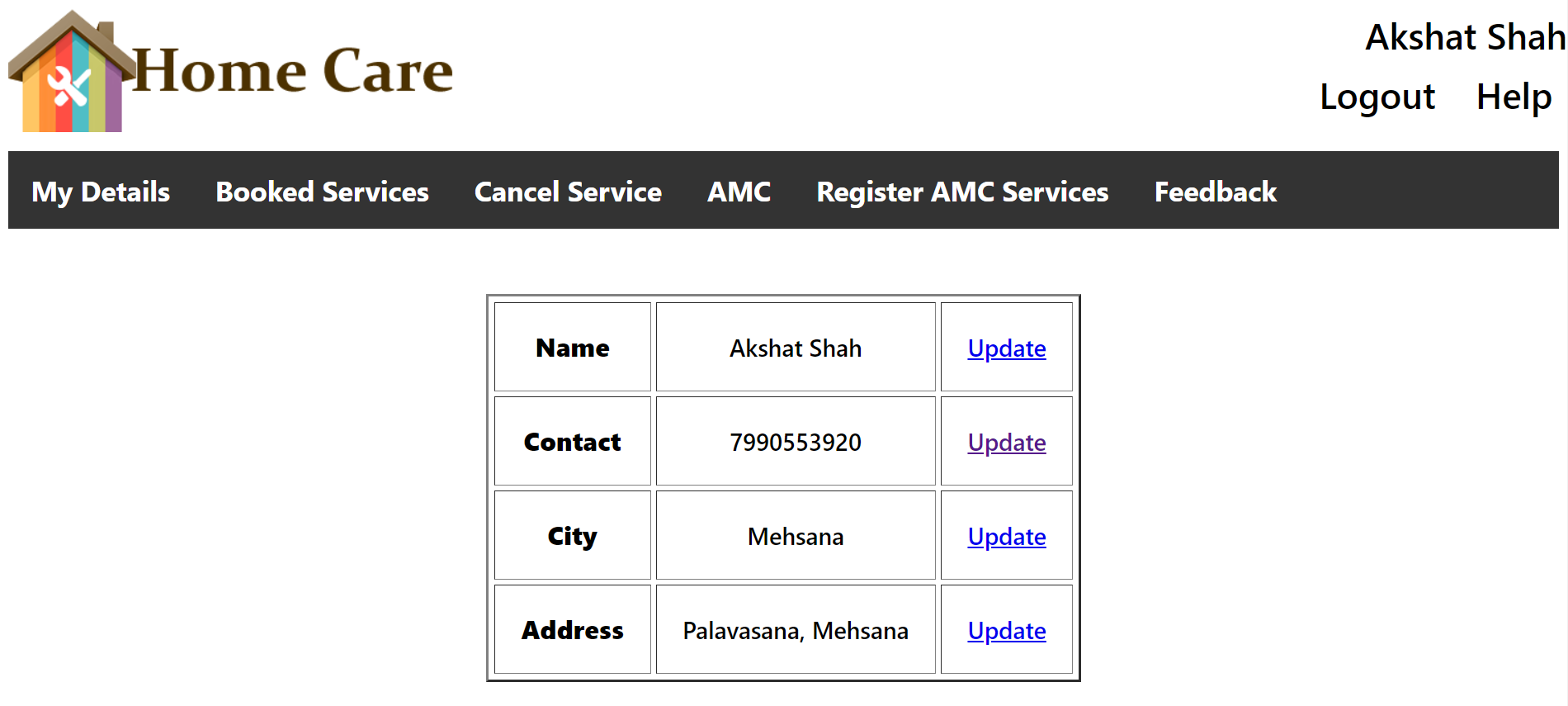
**CLIENT**

**(Procedure-1 used to update the address)**

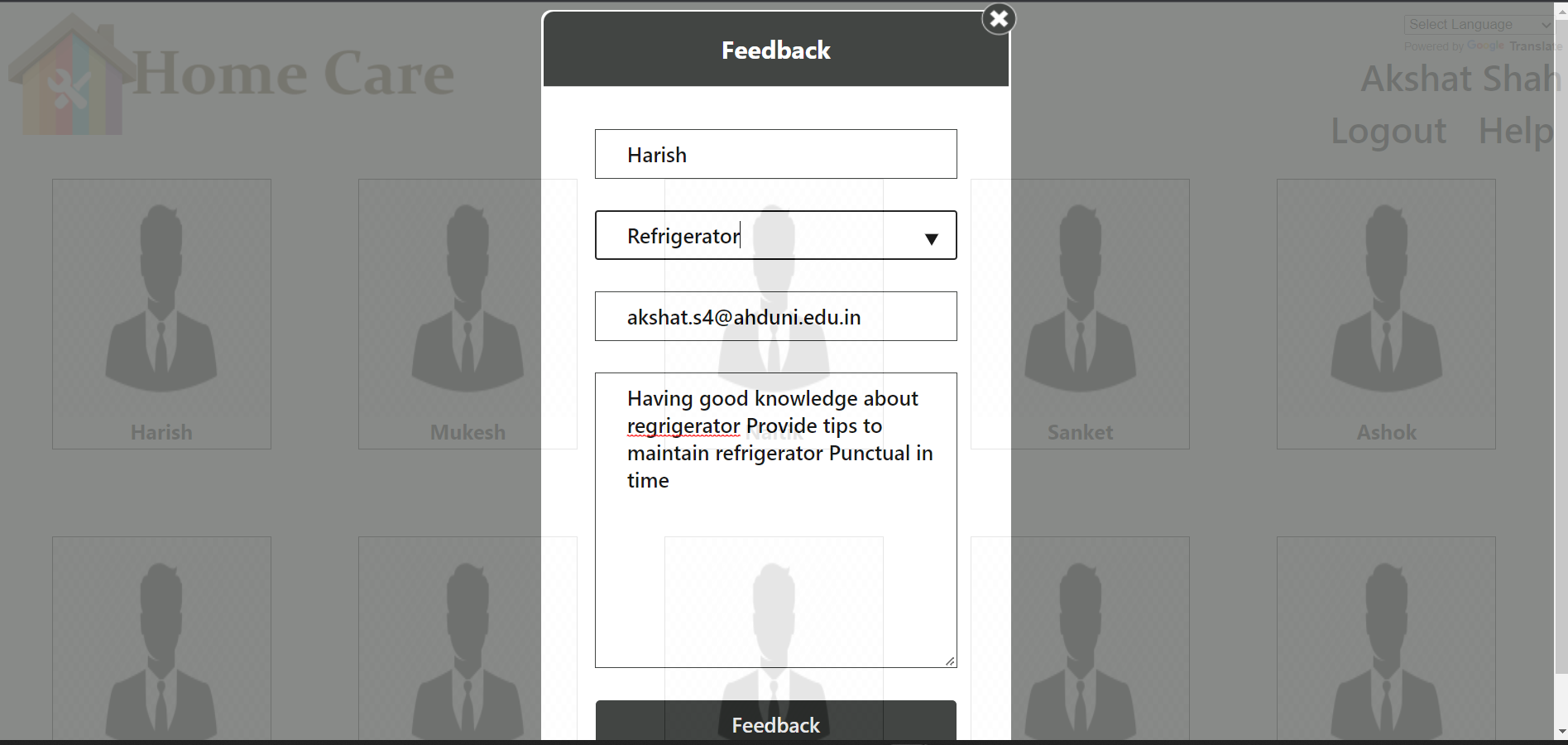
**ADDRESS UPDATE**

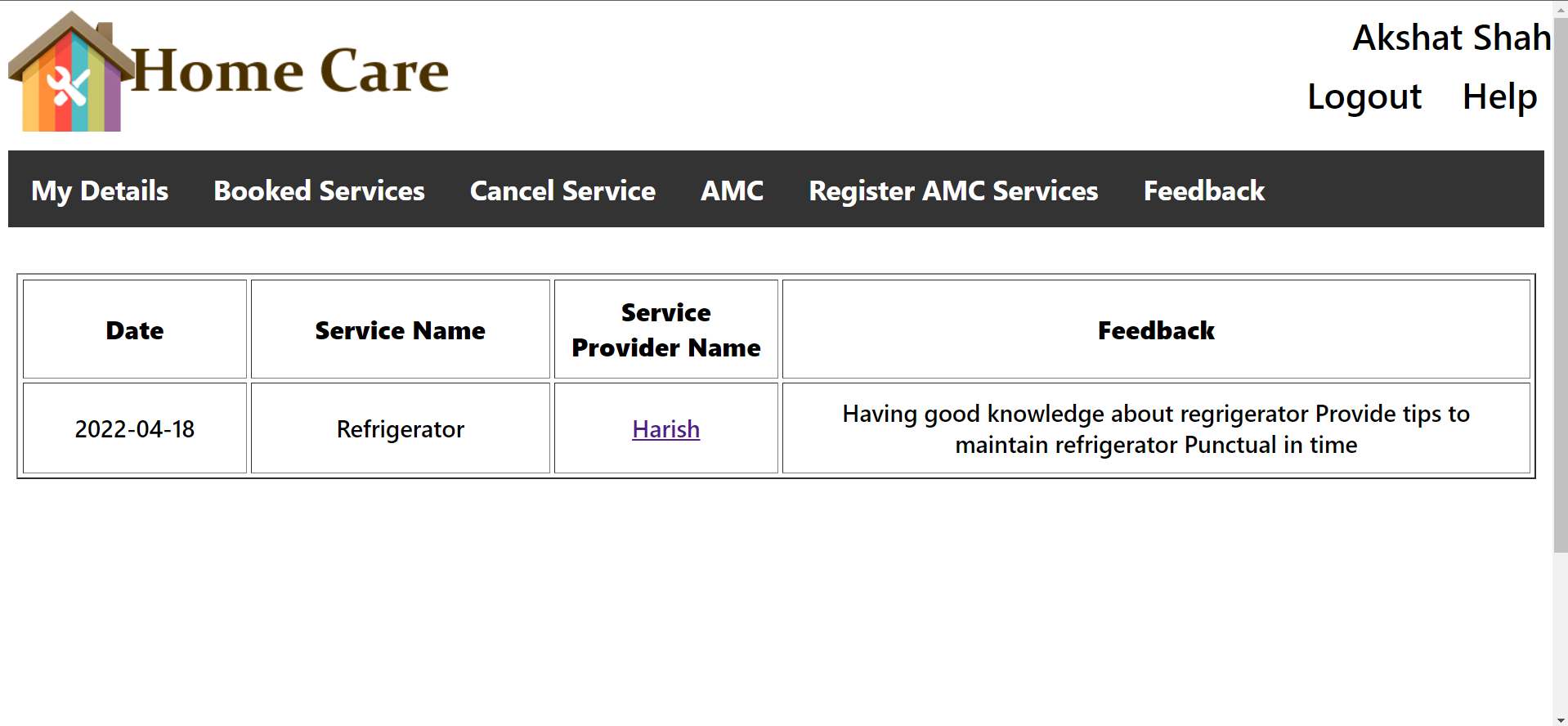
****

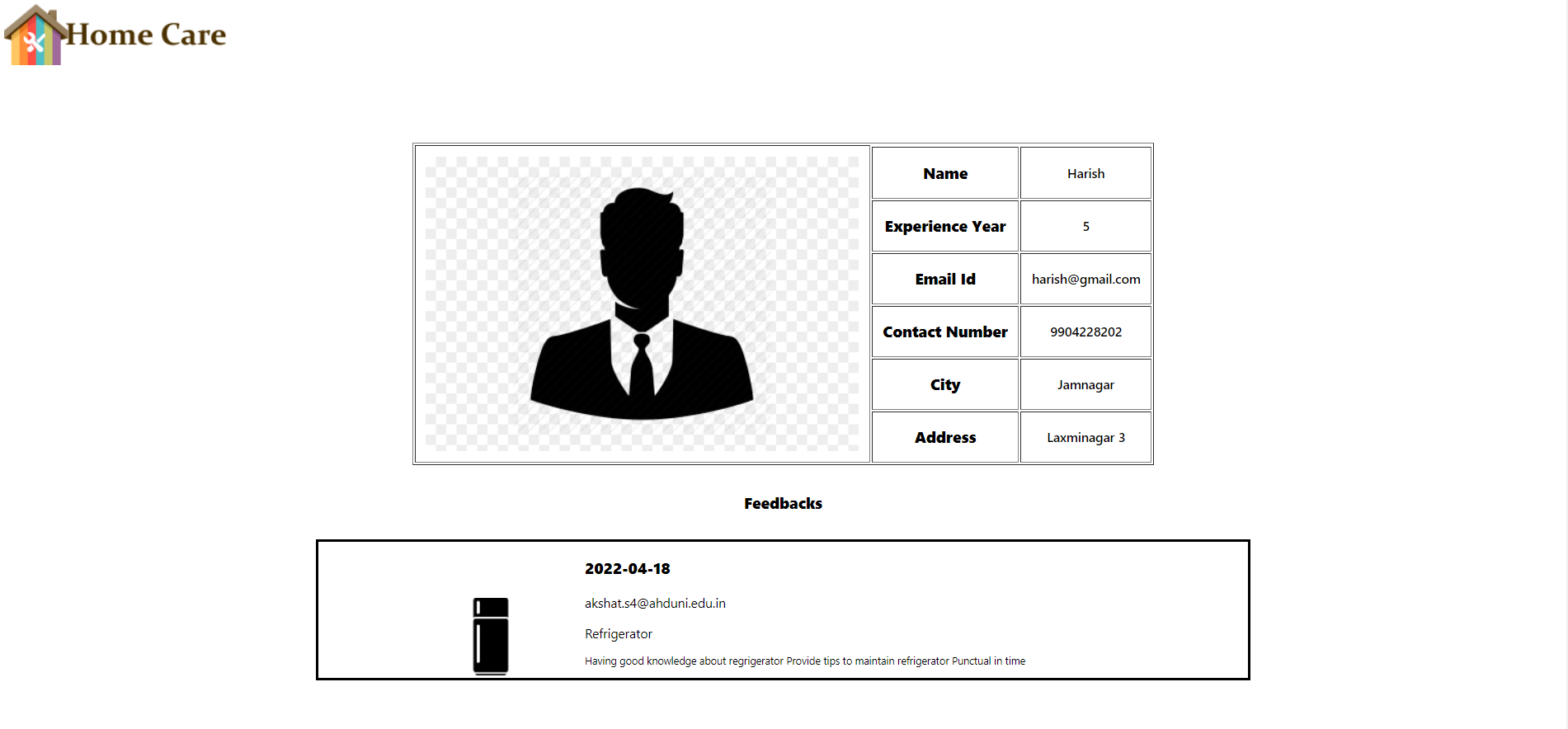
**ADDRESS UPDATED**

****

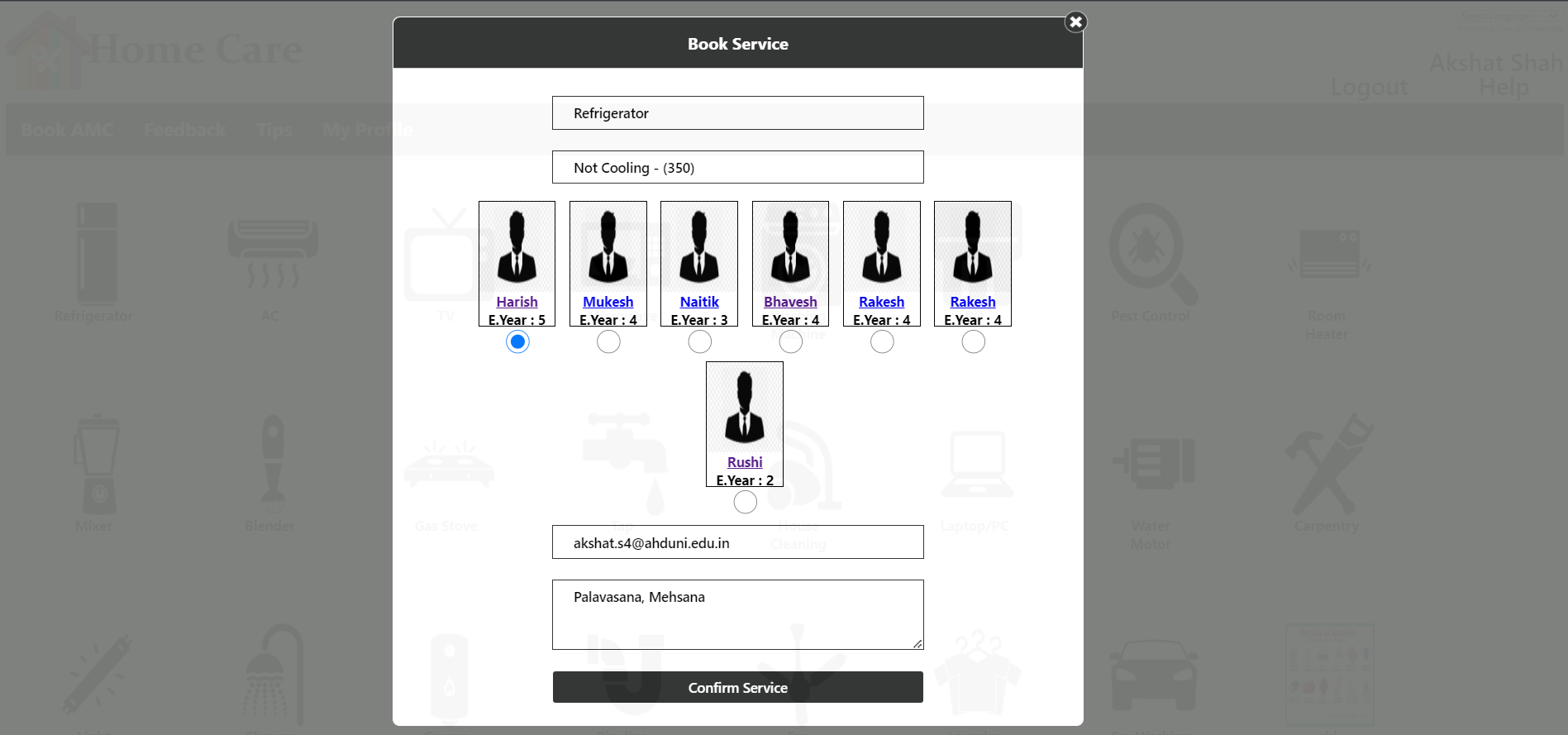
**(Procedure-2 used for giving Feedback)**

****

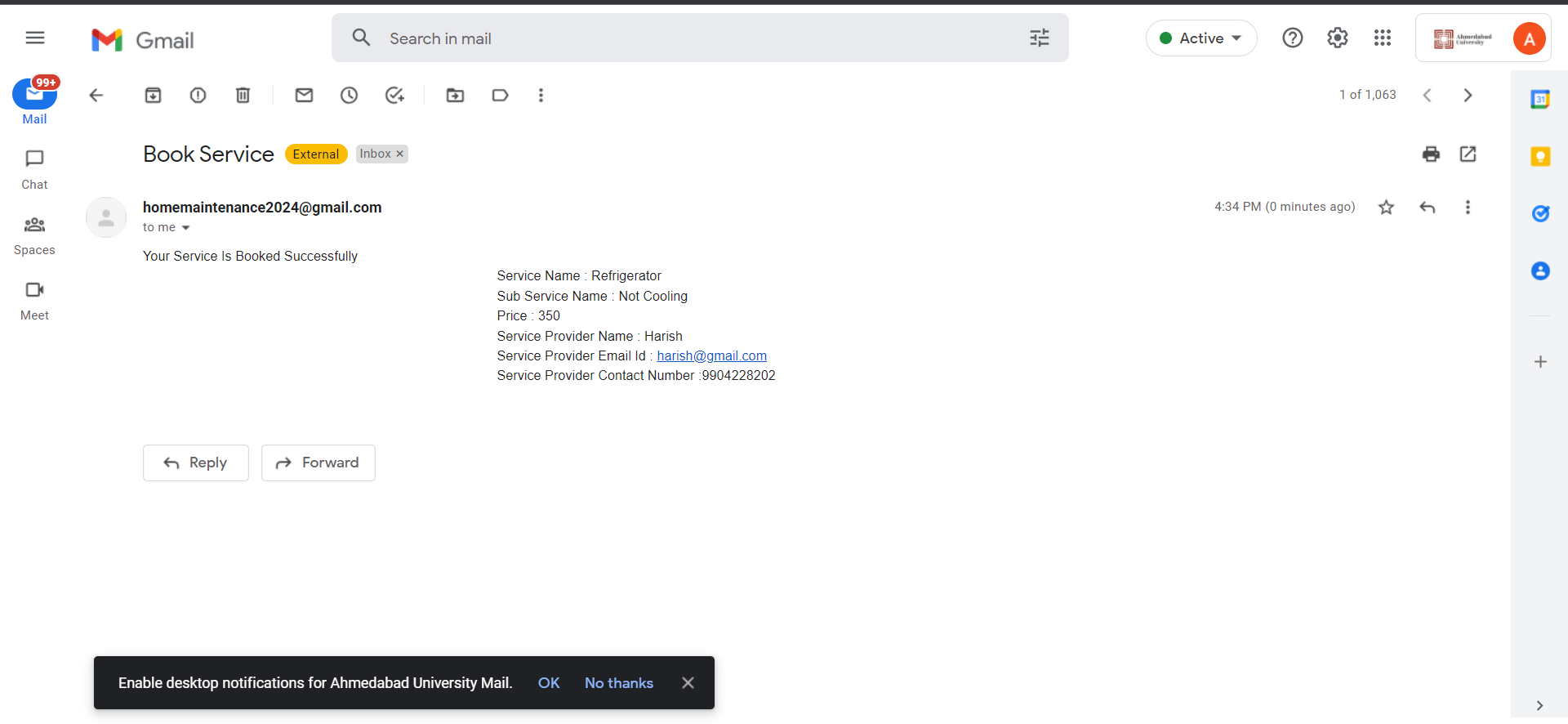
**(Feedback Added)**

****

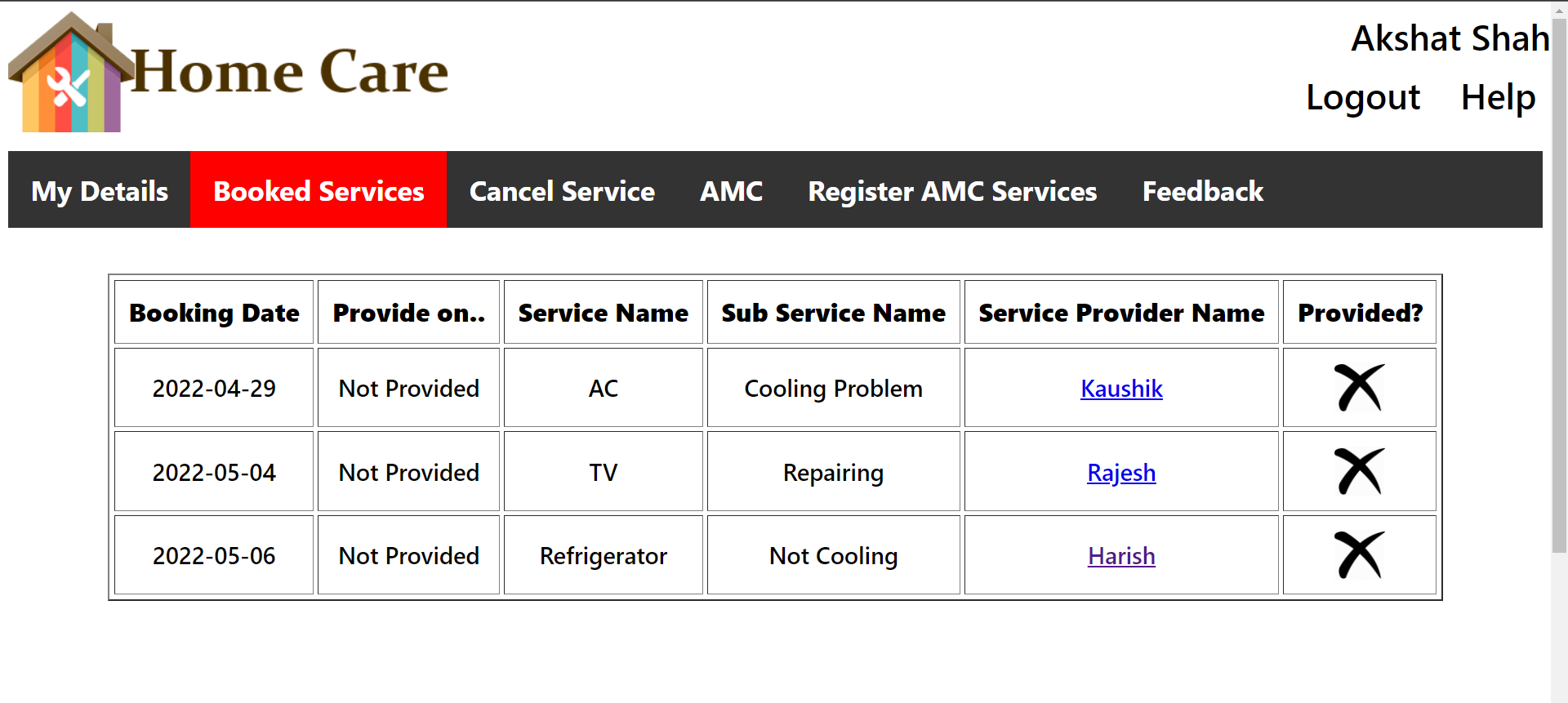
**Book a Service**

****

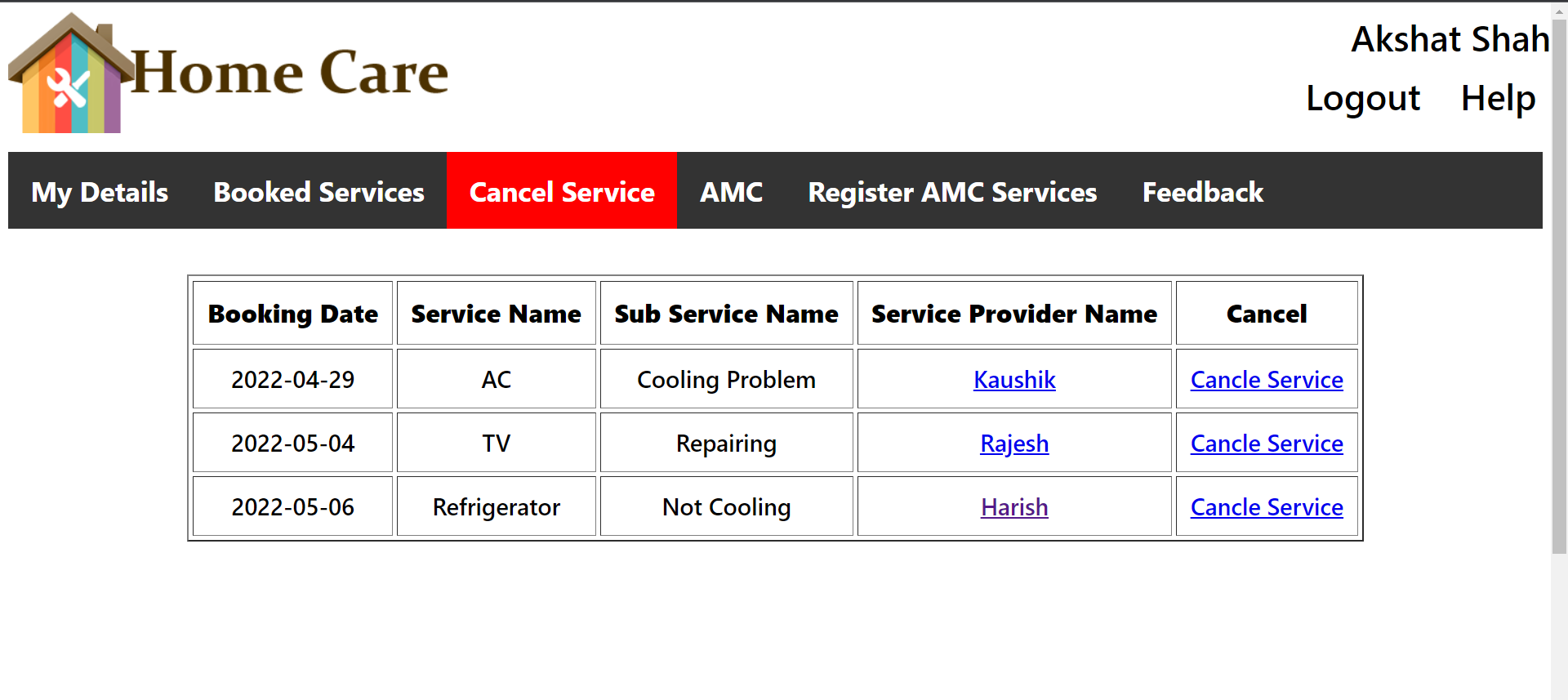
**Confirmation Mail of Service Booked**

****

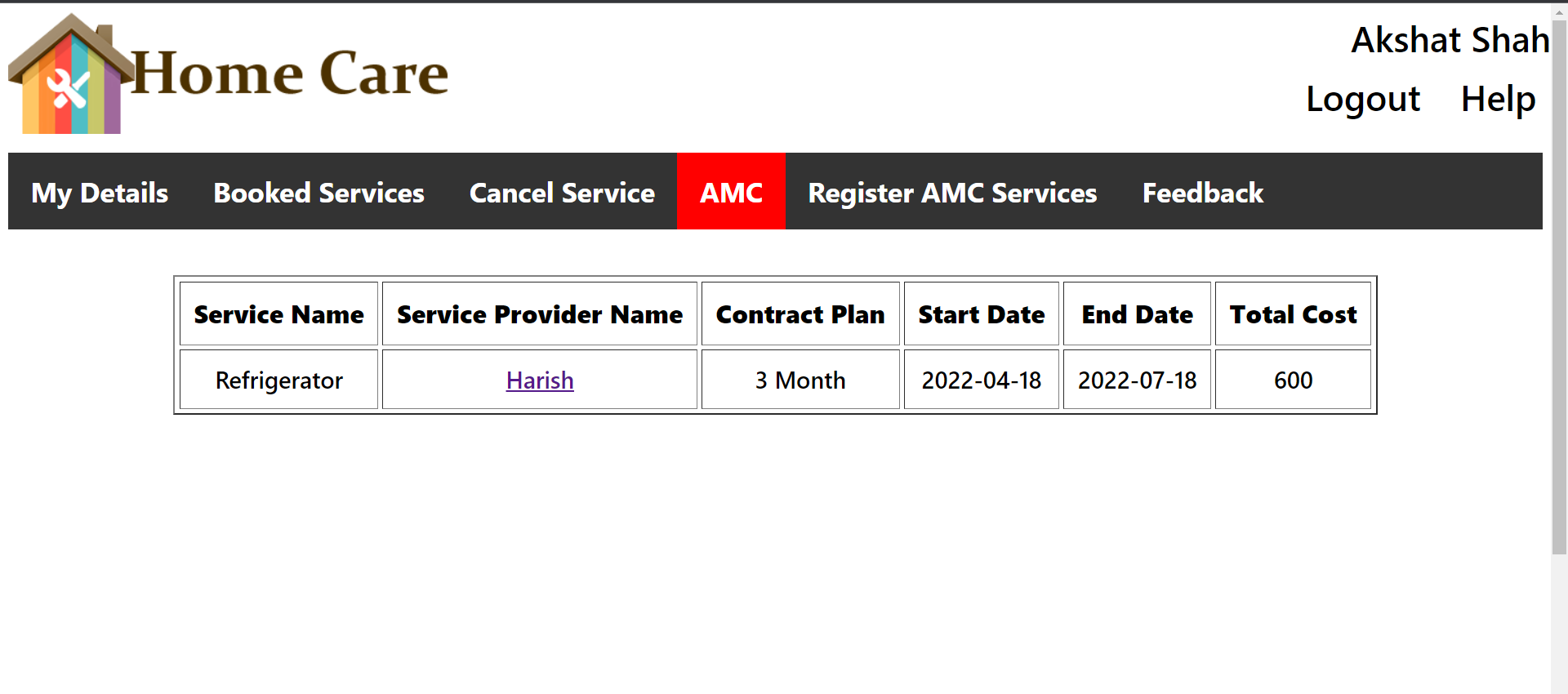
**Services Booked by Client**

****

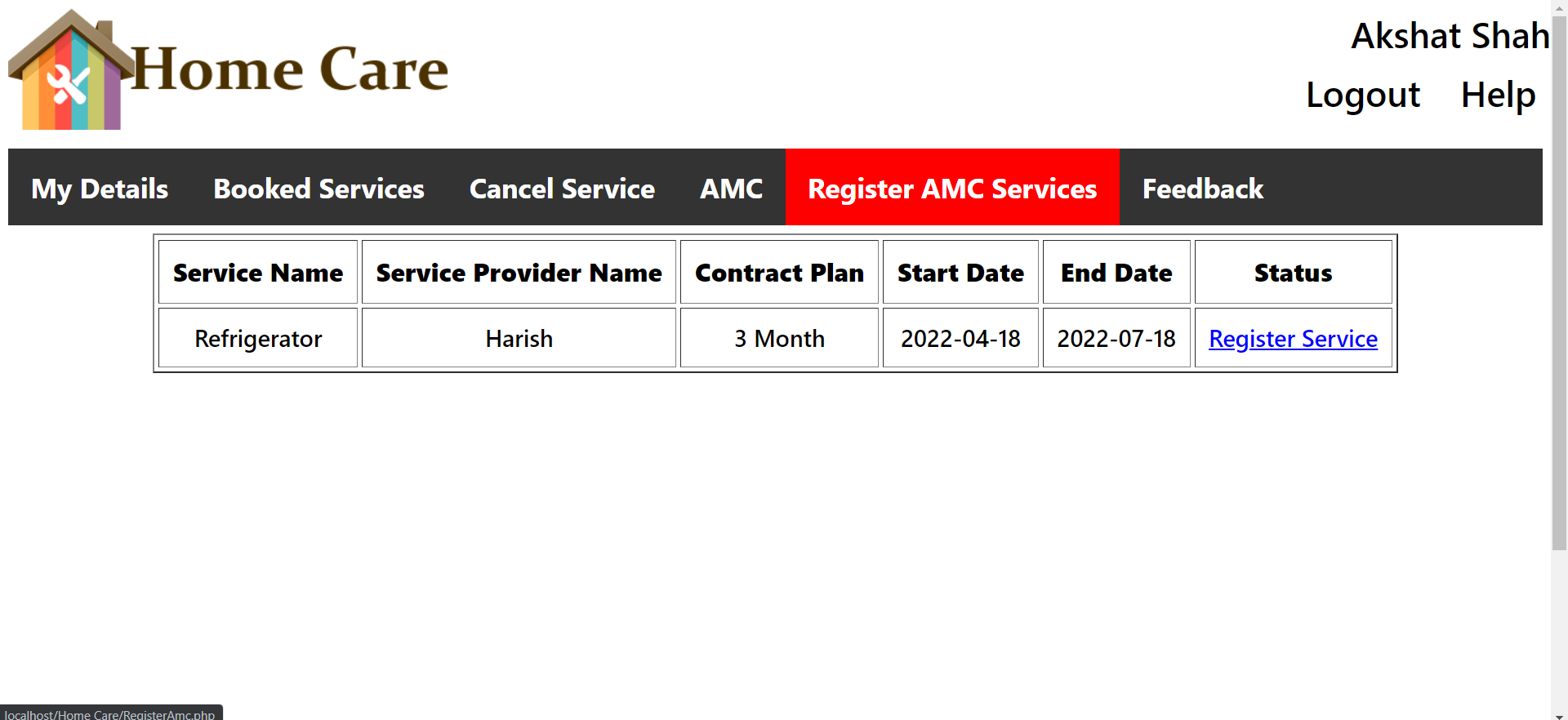
**Services Canceled by Client**

****

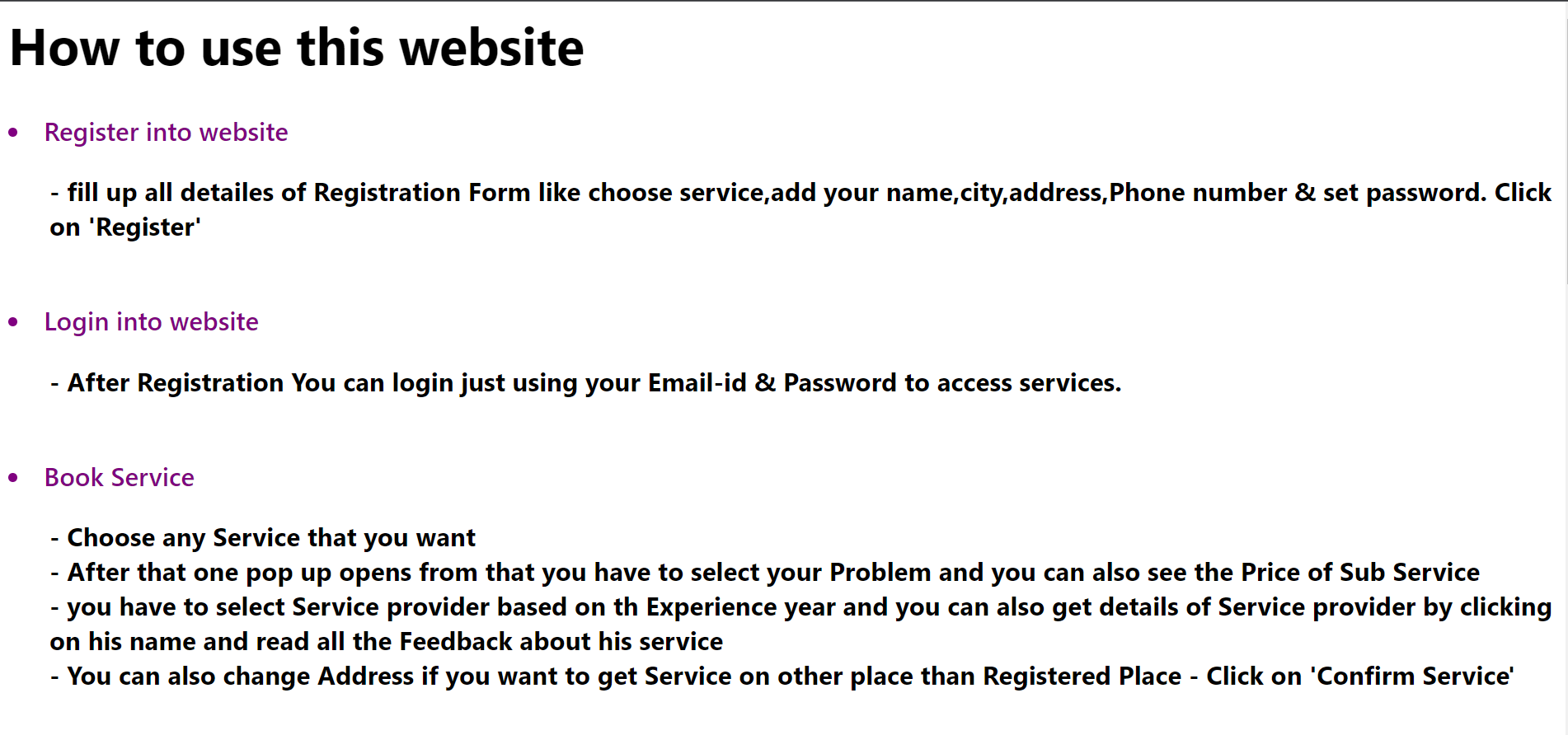
**Annual maintenance Contract**

****

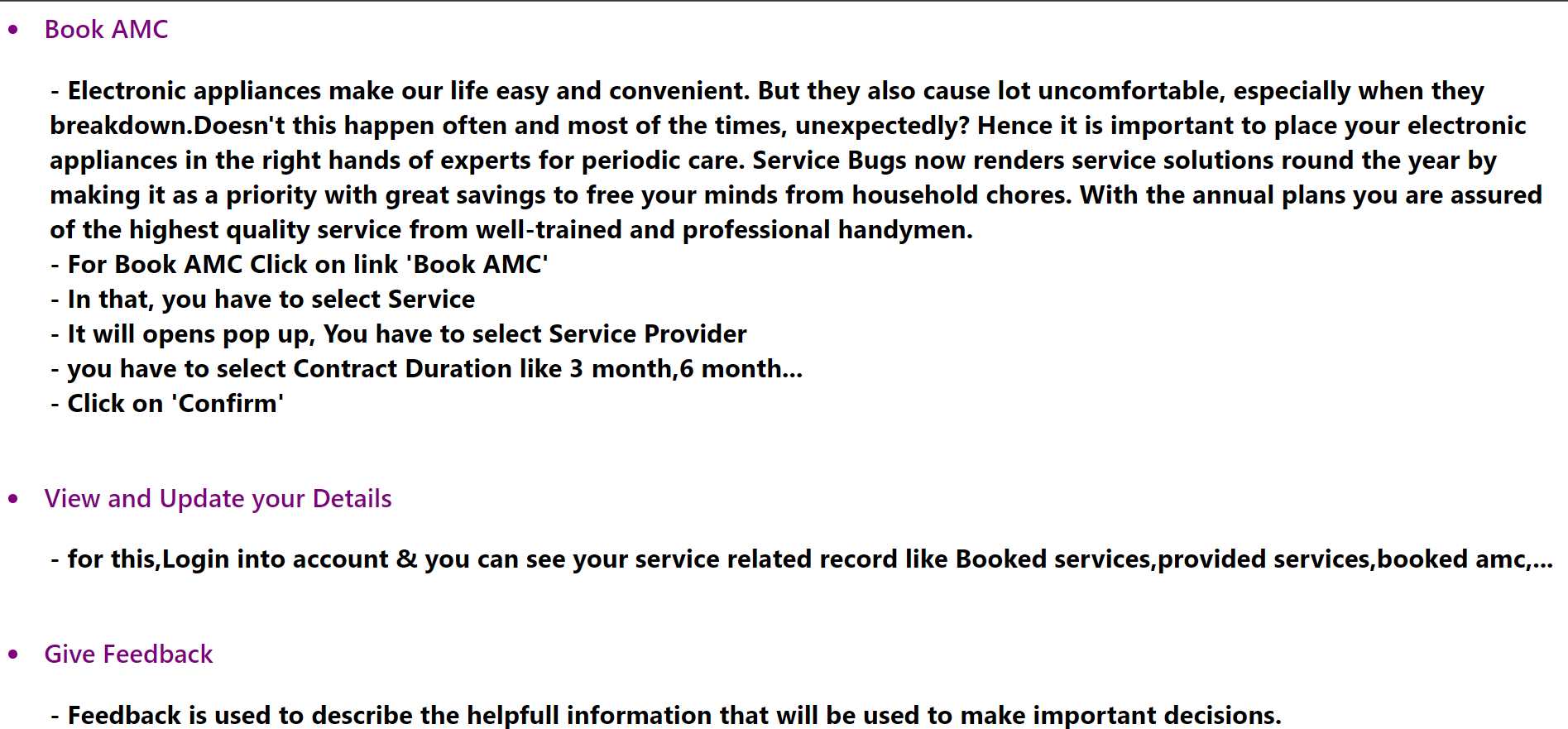
**Booking of Service during AMC**

****

**THE HELP PAGE**







**7. REFERENCES**

* **Fundamental of Software Engineering by Rajib Mall**
* [**www.Brighthubpm.com**](http://www.brighthubpm.com)
* [**www.reqtest.com**](http://www.reqtest.com)
* [**www.Calpoly.edu**](http://www.calpoly.edu)
* [**www.Care4Home.in**](http://www.care4home.in)
* [**www.Housejoy.in**](http://www.housejoy.in)
* [**www.Zimmber.com**](http://www.zimmber.com)