POLITECNICO
MILANO 1863

Computer Science and Engineering

SOFTWARE ENGINEERING 2

A.A. 2015 / 2016

MyTaxiService

Design Document

December 4th, 2015

REFERENCE PROFESSOR:

Mirandola Raffaela

AUTHORS:

Polidori Lucia

Regondi Chiara

Table of contents

1 Introduction

1.1 Purpose

The purpose of the following document is to develop and describe the design of our MyTaxiService software. We have already provided through the development of the RASD document the analysis of the requirements and the description of how our system should work. Now, we are going to describe the software components and the connectors that represent the relations and the interactions between them and the architecture styles and patterns chosen to represent the software architecture of our application.
In few words, this document will explain our decisions concerning the design and the architecture of our system and their justifications.

1.2 Scope

This document is intended to be a detailed design and architectural description of MyTaxiService, for which we have already provided a *Requirements Analysis and Specification Document*.
So, this document provides additional and more detailed information than RASD, and describes our software from a different point of view. For this reason, the Design Document and the RASD document developed earlier have to be both considered in order to have a general view of our project in all its aspects.
The intended audience of this specification include project managers, but also software developers that may use or extend our application in the future.

1.3 Definitions, Acronyms, Abbreviations

We refer to the Glossary provided previously in the RASD document.
In order to make everything clearer and simple, we report here this glossary:

Application: a software made to implement one or more services/activities.

Application domain: set of phenomena and observable properties, typical of the environment in which the software is expected to produce its effects.

Authentication: the act of inserting username and password correctly, in order to complete the log in and access the system.

Code of a taxi: it's the identification number of a particular taxi. Every taxi has a different code.

E-mail notification: a notification sent as an e-mail to a user.

Error message: a notification displayed on the screen used in order to inform the user of an error.

Goal: aim/purpose/utility that has to be reached through the software.

Guest: an unregistered user, or, more in generally, someone that is visualizing the site but has not done the log in yet.

Notification: a notification is something, such as a highlighted box in the system or a specific e-mail received, through which notice is given.

Password: a string that only the registered user knows and that is necessary for the authentication.

Personal profile page: the registered user's private page, where he can see his notifications, choose whether to request or reserve a taxi and click on specific links in order to visualize his personal information and the list of all his reservations.

Registered user: an user that is already registered to the system and is allowed to do the log in.

Request: the act of call a taxi through the web application. After a request, a taxi will be allocated and will arrived at the required location as soon as possible.

Requirement: a requirement express a desire of the costumer concerning the application. Every requirement has to be satisfied in order to reach all the goals.

Reservation: the act of book a taxi from a point A to a point B, on day D, at the time X.

Taxi-driver: the person who drives the taxi and use the application that is installed on board.

User: see "Registered user".

Username: a string, chosen by the registered user, to identify himself and that is necessary, together with his password, for the authentication.

Web application: an application accessible through the web using a network (such as an intranet or the Internet).

In addition to the vocabulary we reported above, we add some new terms that we are going to use in this Design Document:

Application server: a server program in a computer in a distributed network that provides the business logic for an application program.

Architectural style: a particular pattern that focuses in the large-scale of a system. A style is what characterizes an architecture with respect to another, imposing a series of constraints.

Client: a piece of computer hardware or software that requests and accesses one or more services made available by a server usually (but not always) located on a different computer system.

Component: a part, piece or subsystem that performs a distinctive and necessary function in the operation of a system.

Component diagram: a graphical representation of how the components of a system are connected in order to form larger components or software systems.

Database server: a computer program that provides database services to other computer programs, following the rules defined by the client-server paradigm. It performs tasks such as data analysis, storage or data manipulation.

Deployment diagram: a graphical representation that models the hardware components of a system, how they are connected to each other and what software components run on each identified node.

Server: a computer program or a machine that shares data or resources among a group of clients. It offers a set of services, waiting for requests from one or more clients and responding to them.

Tier: a physical level in the architecture of a system. It's an hardware level (a machine) where software layers are installed.

1.4 Reference documents

We wrote this document referencing the *Requirements Analysis and Specification Document* (*RASD*) we have already developed.
As for the organization of the chapters and the paragraphs, we followed the structure and the subdivision into paragraphs of the template for the design document made available by our professor. Finally, we also helped ourselves consulting the IEEE Standards for the Design and the Architecture Descriptions.

1.5 Document structure

Our Design Document is divided into eight main parts:

- *Section 1*: Introduction
  this first section gives an introduction to the Design Document, specifying the purpose and the scope of the document, the glossary (containing all the terms, definitions, acronyms or abbreviations used) and the documents we referred to in order to develop this paper.
- *Section 2*: Architectural Design
  this section is the core of the document. It gives information about the design and the architecture of our application, defining all the software and hardware components characterizing the system and how they can interact between each other. Finally a brief description of the architectural styles and patterns selected is given, in order to explain how and why we decided to use them.
- *Section 3*: Algorithm Design
  this section contains the description of two of the main algorithms used in order to execute specific operations and functionalities of our system.
- *Section 4*: User Interface Design
  this section contains a detailed representation of the users interfaces. Through specific and very detailed mockups, the interfaces of the web application for the users and the mobile app for the taxi drivers are illustrated.

- *Section 5*: Requirements Traceability
  this section contains the correspondences between the requirements we have listed in the RASD document and the components we have identified during the design and architectural analysis.
- *Section 6*: Used Tools
  in this section, all the tools we have used in order to develop this document are listed.
- *Section 7*: Working Hours
  this sections contains the result of our effort, quantified in the number of hours we have needed in order to develop this document.
- *Section 8*: Appendixes
  in this section we have an appendix where all the modifications we have done to the RASD during the development of this document, are described.

2 Architectural Design

2.1 Overview

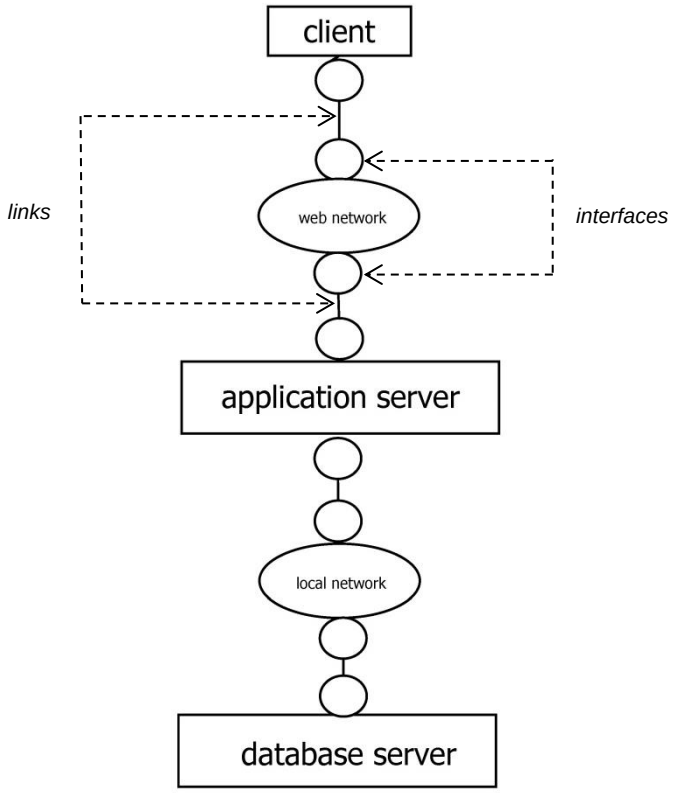
In this chapter we are going to describe the architectural choices that we made for our system. Starting from a general description of the high level components and their interactions, then we will describe through a deeper analysis all the components of our system and the physical relation between software and hardware components. In this way we can show not only which components characterized our system, but also on which hardware component of a distributed system each one of the software component is located or executed.

Later, using some explicative sequence diagrams, we will describe the way our components interact in order to accomplish specific tasks (typically the ones related to the use cases we have already provided in the RASD document).

Finally, we will explain which architectural style and patterns we chose, how and why we have decided to use them.

2.2 High level components and their interaction

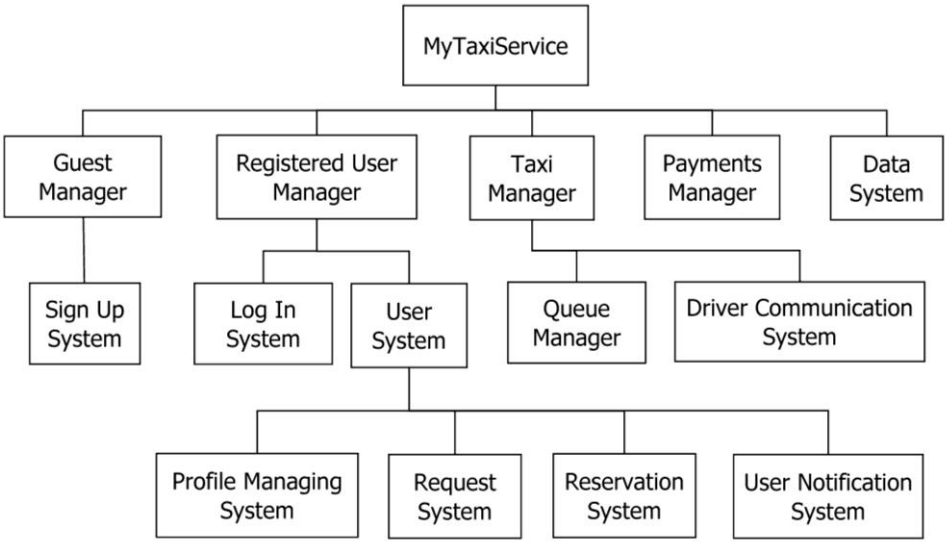
Using the graph below to make everything clearer, we want to give an idea of our software architecture. On each server, different components (sub-systems), specific for each functionality, are installed, managing different tasks and operations.



2.3 Component view

We thought about our system as composed by different sub-systems (components) in order to simplify the management of the functionalities. In particular, each component can be seen as independent from the others and have an own logic specific function. Obviously, each component can communicate with the others in order to execute its operations and satisfy all its tasks.

We now provide below a simple hierarchical schema representing the sub-systems that characterized MyTaxiService:



Guest Manager: this module is interfaced with the guest client and its job is to manage all the guests of the application, showing the homepage and all the pages reachable from the homepage that all the visitors can navigate. It contains another sub-system for the sign up functionality.

Sign Up System: it's the sign up sub-system. It's contained in the Guest Manager component and it's the module that manages the whole registration process. It can communicate with the Registered User Manager and the Data System to notify the system about the creation of a new user (showing him his new profile) and to add his data into the database.

Registered User Manager: this module is interfaced with the registered user client and its job is to manage all the registered users of the system. It contains other modules, one for each user functionality, and communicates with the Data System in order to store or retrieve the information needed.

Log In System: this is the sub-system that manages the whole log in process. He has to provide functions for the username/password verification and for the password recovery. He has to show the correct profile to the user who's just logged in.

User System: this module of the Registered User Manager manages all the functionalities related to a registered user. It contains four other modules for performing all the possible user's functionalities.

Profile Managing System: this module has to provide the basic functions to manage personal profiles (show profile, show personal information, update personal information, show user's notifications, change profile pictures and background images…).

Request System: this module manages all the users requests. It has to provide functions to create new requests, store requests information and communicate with the Taxi Manager to check taxis availability and allocate a taxi for each request.

Reservation System: this module manages all the users reservations. It has to provide functions to create or delete reservations, store reservations information and communicate with the Taxi Manager.

User Notification System: this module manages the whole notification process. It has to provide functions to send notifications (by e-mail or on the profile) to each user and communicate with the Request and Reservation Systems and with the Payments Manager, because it is also in charge of sending e-mails to confirm a payment or containing penalty fees to be paid in the following months.

Taxi Manager: this module is interfaced with the taxi driver client and manages all the taxi drivers. It has to provide a GPS location function, in order to accurately localize the distribution of the taxis, functions that allow the communication with the driver and to send or receive drivers notifications. It communicates with the User System in order to manage requests and reservations.

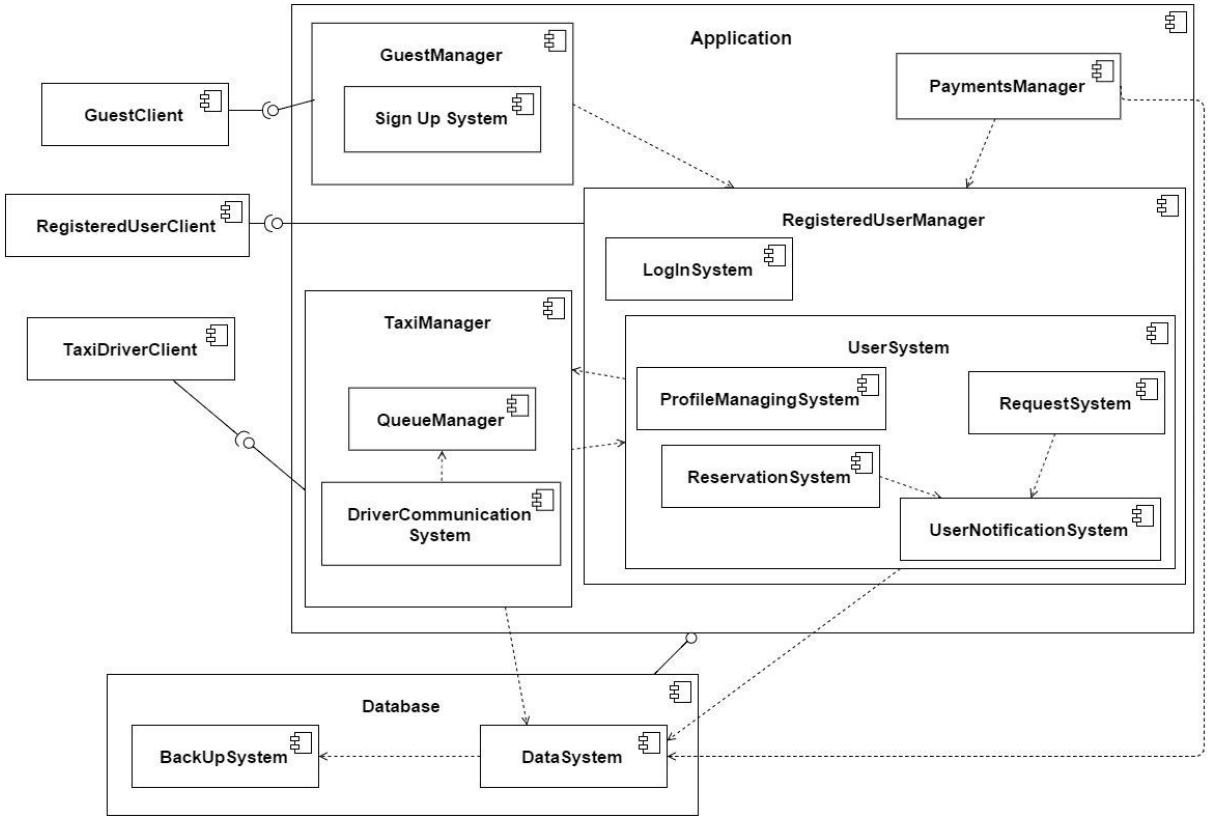
Queue Manager: this module has to provide functions to manage all the taxis queues of the city. It has to communicate with the Driver Communication System in order to retrieve information about taxi's availability and eventually move it to the end of the queue.

Driver Communication System: this module manages all the drivers notifications. It has to provide functions to send and receive notifications from the taxi drivers, in order to have information about their availability and to forward requests that they have to take care of.

Payments System: this module has to provide functions to manage all the payments processes and to check all the payments associated to each request or reservation. It communicates with the Data System to store payments information and to check all the requests and reservations and with the User Notification System for e-mail notifications containing payments confirmations or penalty fees for each incomplete payment.

Data System: it contains all the application data in a relational database; it has to provide functions to manage data (store and retrieve data from the database when required).

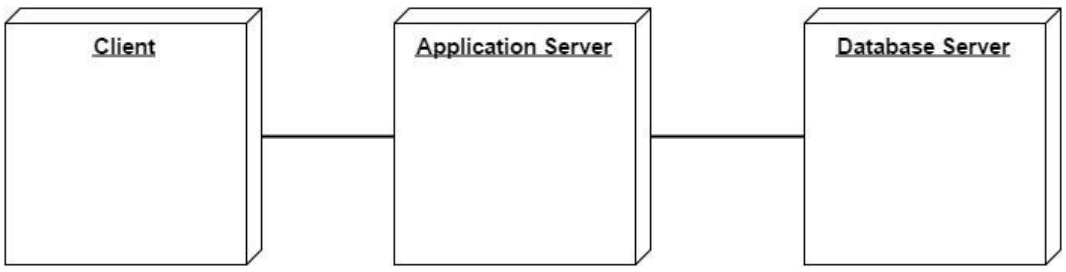
Here, a *component diagram* that clarify everything we have just describe: in this diagram all the components of our system and their interactions are represented.



2.4 Deployment view

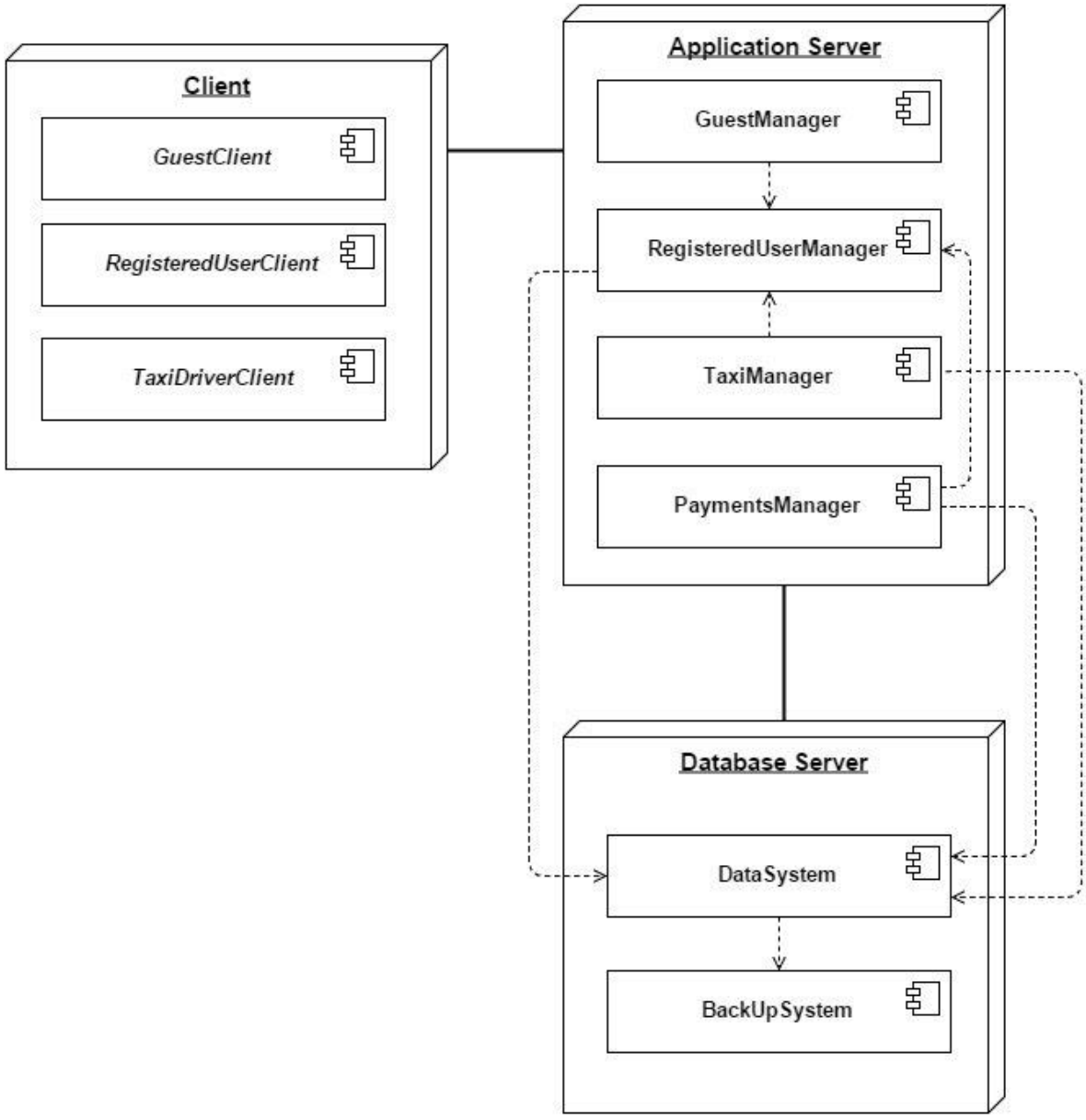
In this paragraph we describe, using a simple *deployment diagram*, how the components of our system are mapped to hardware components, representing on which hardware component each of our software components is located or executed.

Each node of the diagram represents “something” on which a software component can be located: usually it represents an hardware component (device), such as a sensor, a server or a mainframe. Two nodes can be connected graphically using a segment that represents a communication path (connection) between the two components.



It is possible, and usually very useful, to integrate the component and the deployment diagram, in order to make more understandable the distribution of the different components on the nodes of the system.
In our case, we have on the Client the client components, one for each actor involved in our system. The majority of the components are situated on the Application Server, that represents the core of the application. Here is where all the commands are received, interpreted and executed, all the tasks are satisfied and where the whole application is managed and coordinated.
On the Database Server there are the components involved in the data management: the data system, containing all the main data and information of the application, stored in a relational database, and the back-up system, containing and managing a copy of the primary database, updated automatically by the system.

Reported below it's the integration between the two diagrams proposed in this paragraph and the previous one:

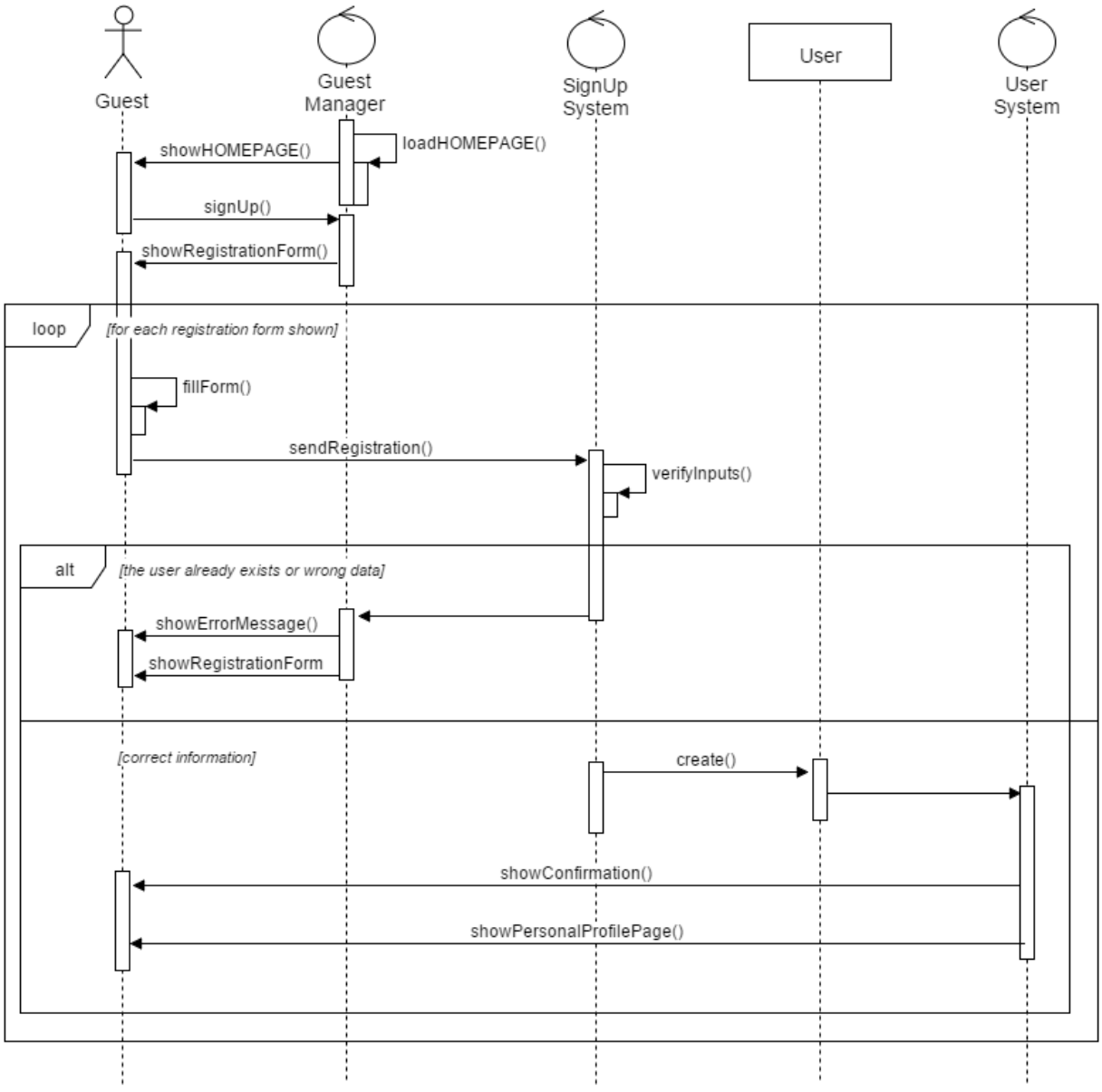


2.5 Runtime view

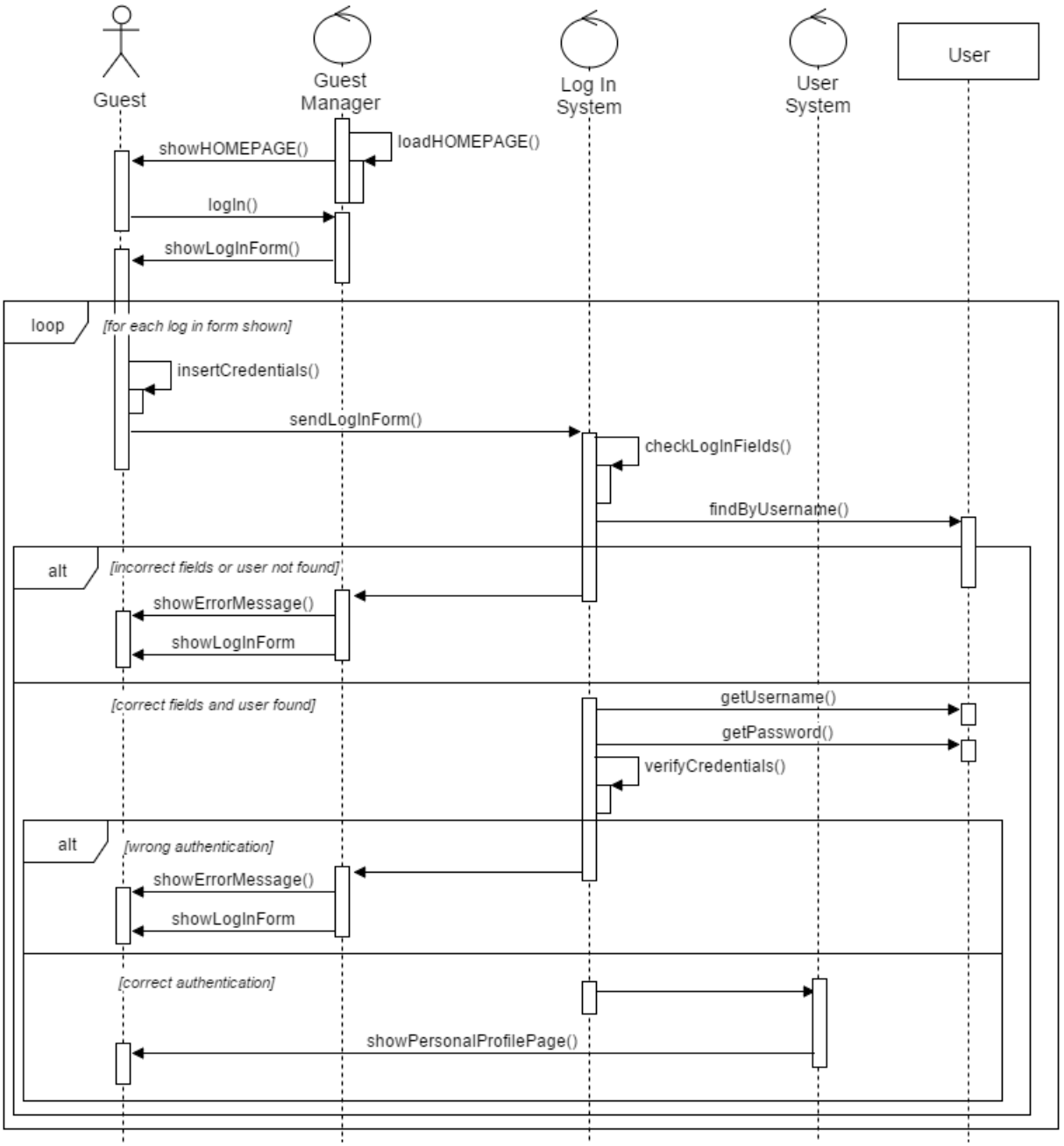
In this paragraph, following the component and deployment view previously illustrated, we are going to describe how the components of our system interacts with each other and with the actors of our system, in order to accomplish specific tasks related to the use cases we have identified in the RASD document.

In order to do that, we provide some sequence diagrams that better explain all the operations executed and the interactions between the different components of our application.

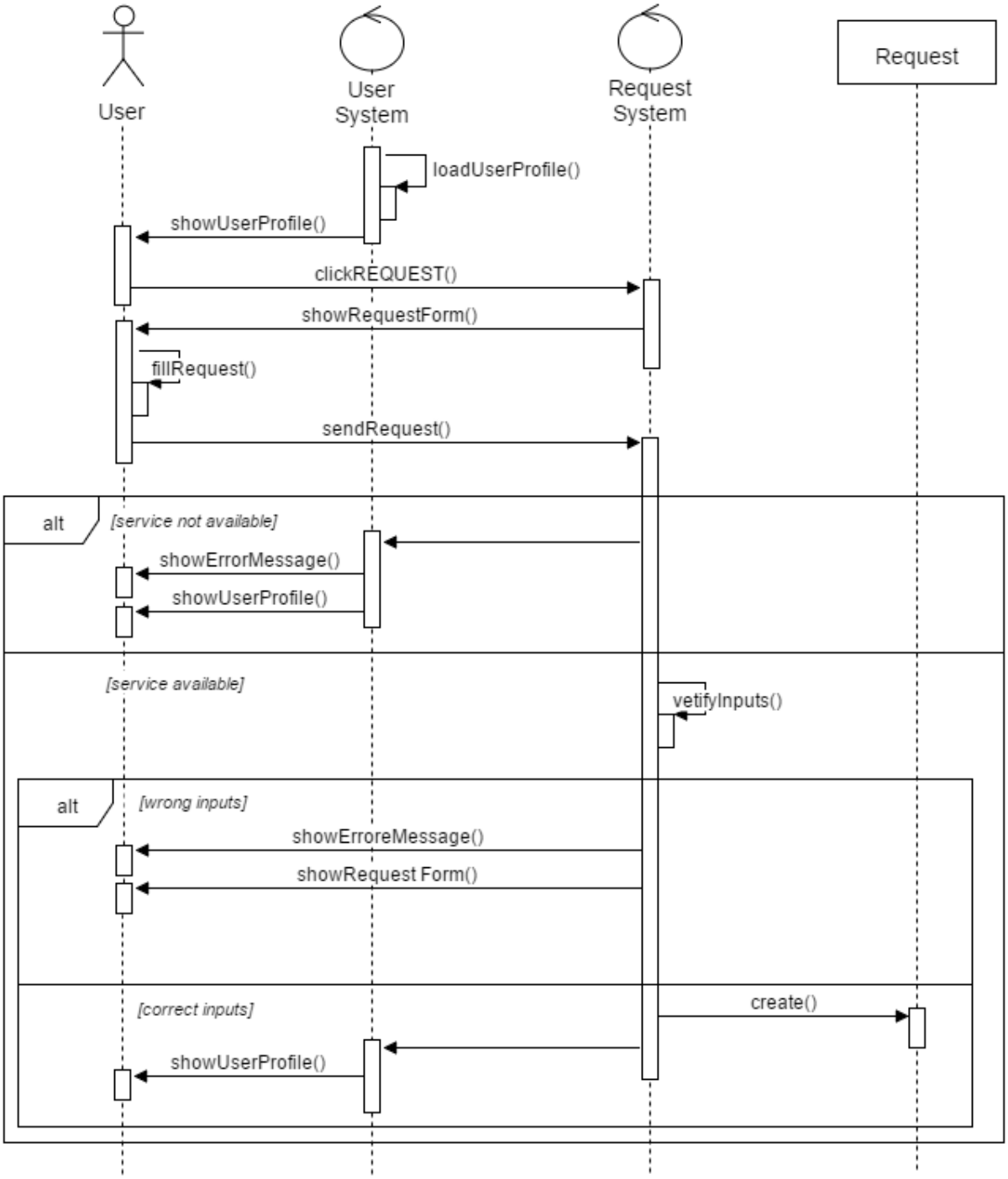
2.5.1 Sign up



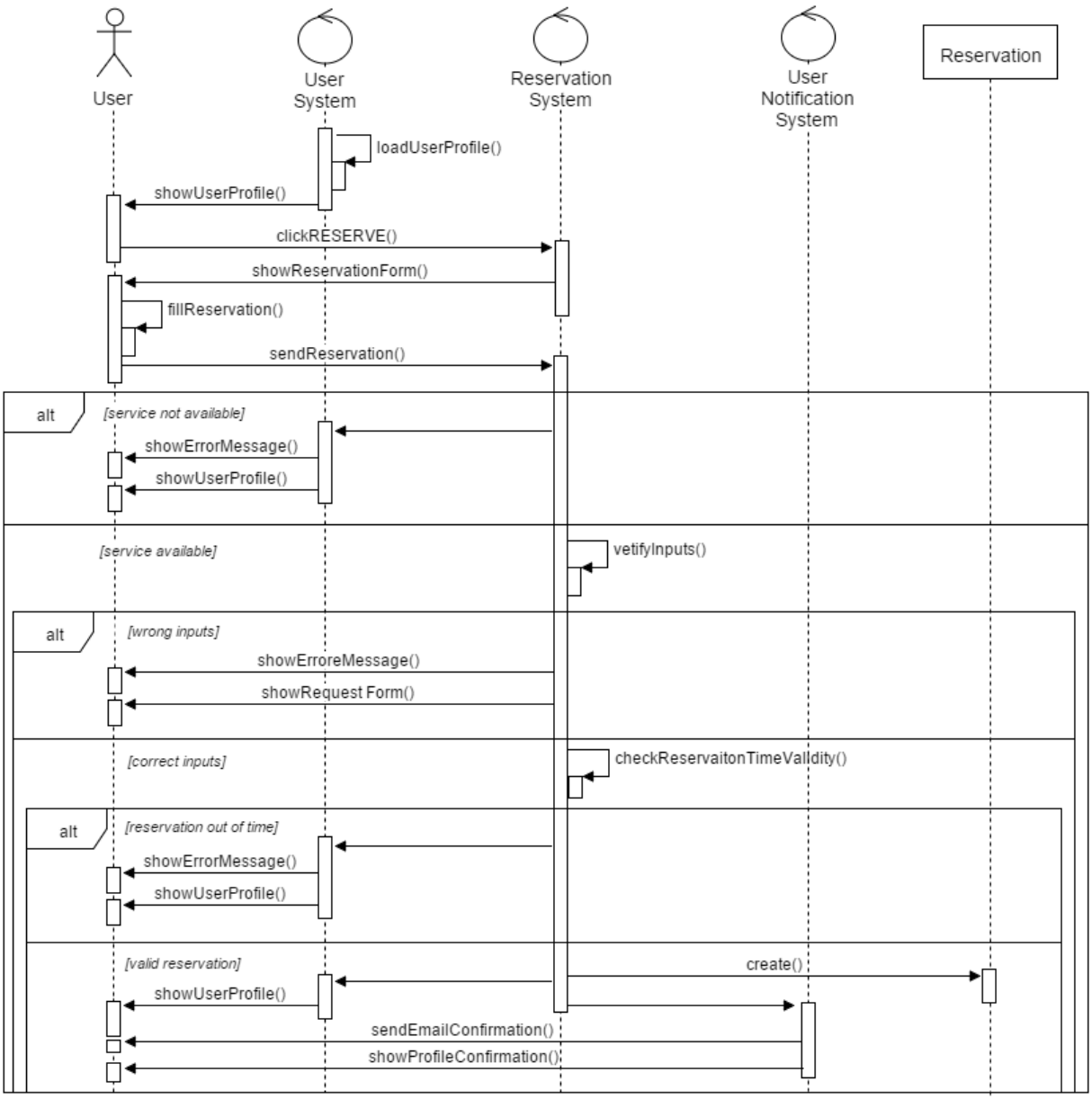
2.5.2 Log in


Guest
Guest Manager
Log In System
User System
User
showHOMEPAGE()
loadHOMEPAGE()
logIn()
showLogInForm()
loop
[for each log in form shown]
insertCredentials()
sendLogInForm()
checkLogInFields()
findByUsername()
alt
[incorrect fields or user not found]
showErrorMessage()
showLogInForm
[correct fields and user found]
getUsername()
getPassword()
verifyCredentials()
alt
[wrong authentication]
showErrorMessage()
showLogInForm
[correct authentication]
showPersonalProfilePage()

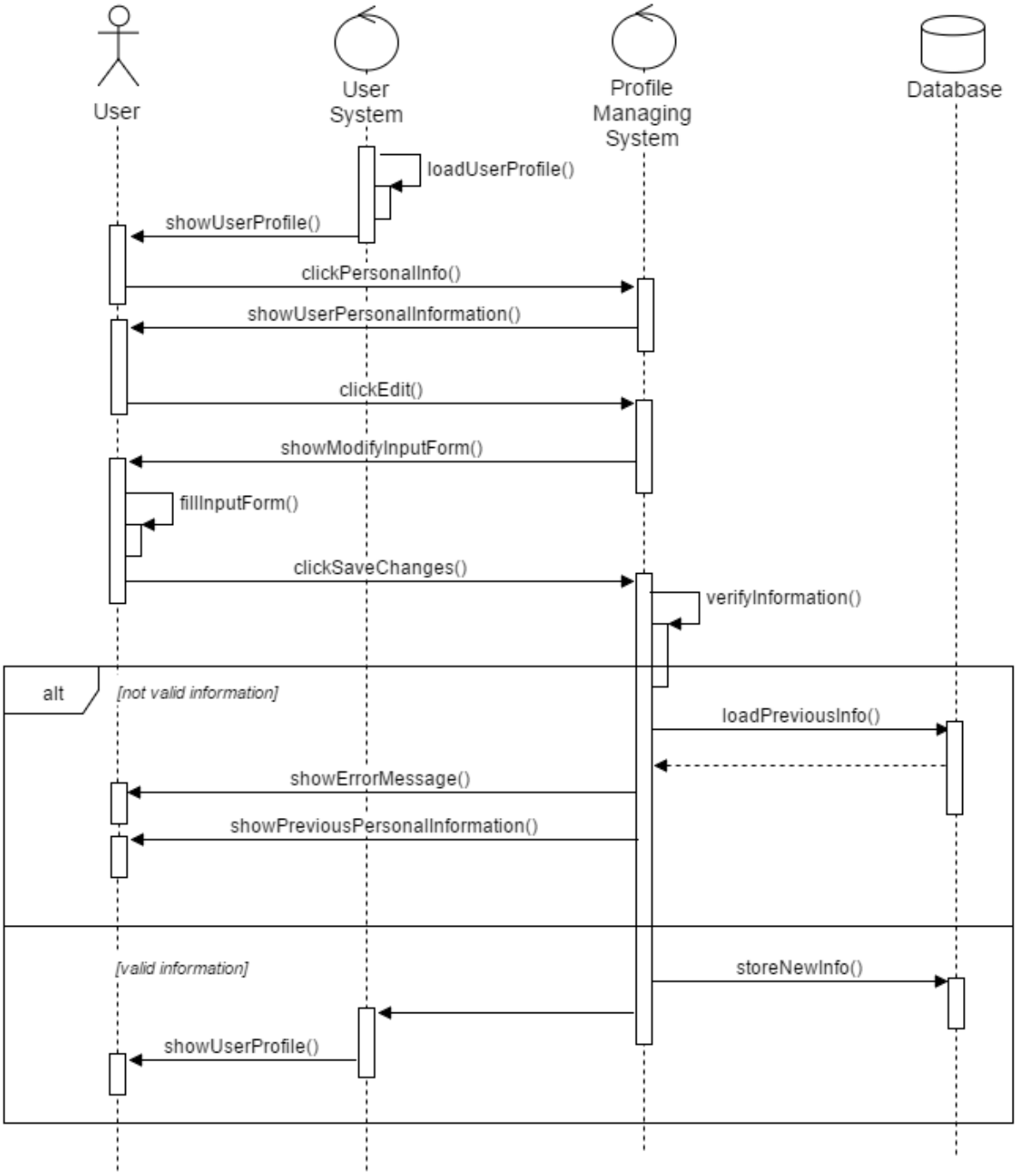
2.5.3 Request a taxi



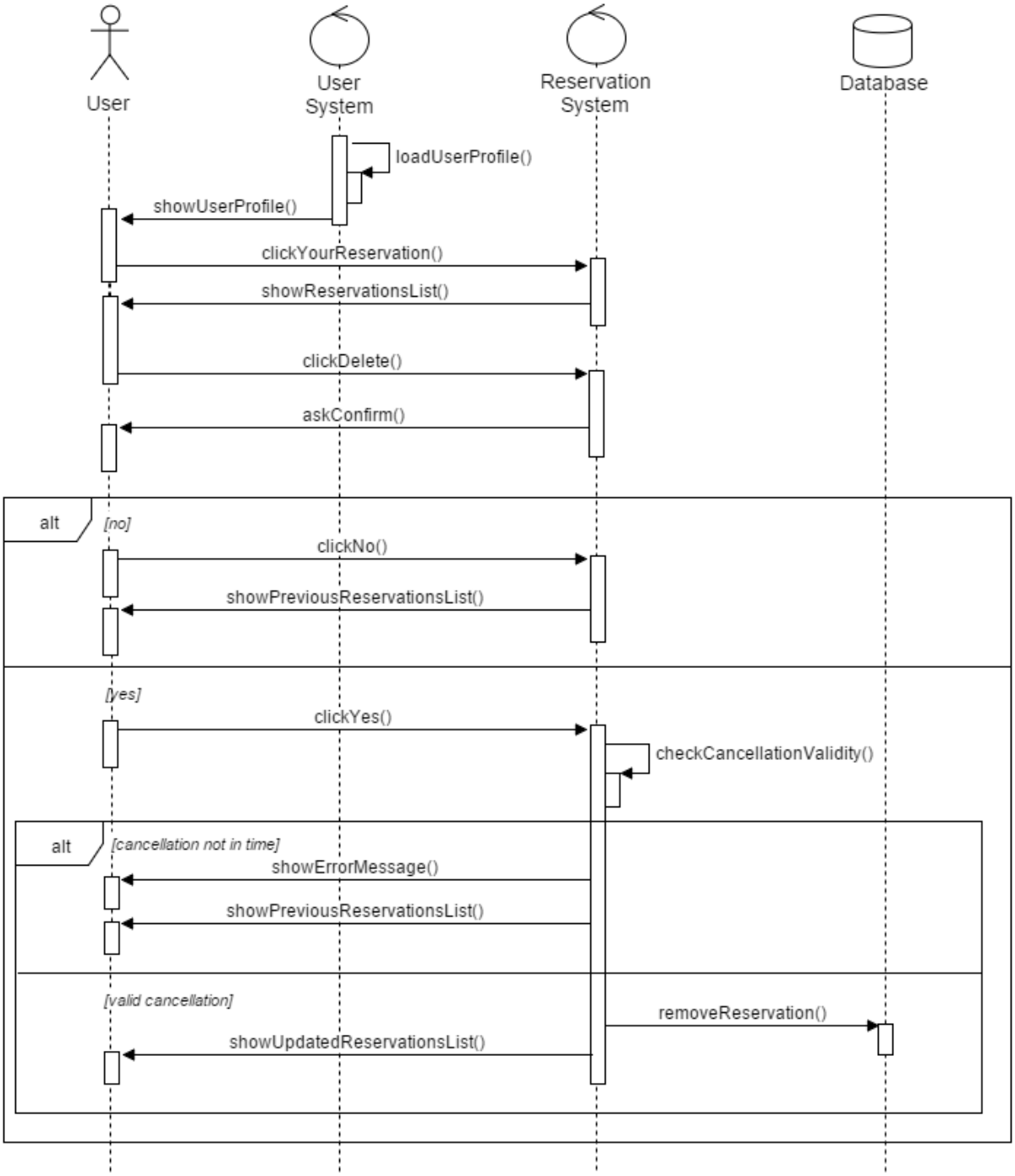
2.5.4 Reserve a taxi



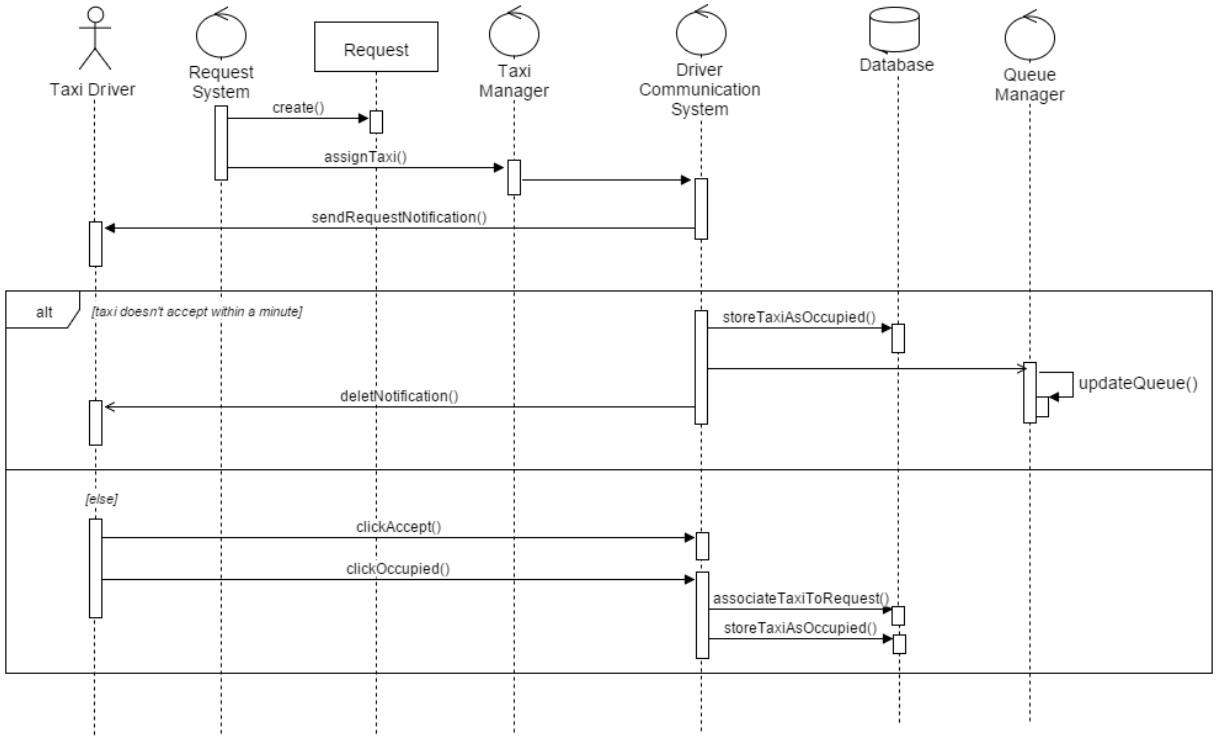
2.5.5 Update personal information



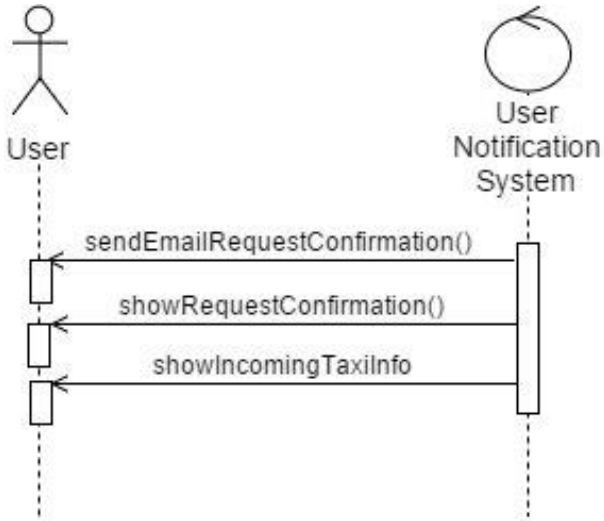
2.5.6 Delete a reservation



2.5.7 Taxi driver request communication



2.5.8 User request confirmation



2.6 Component interfaces

We now try to explain more deeply how the different components of our architecture are interfaced with each other. The client communicates with the server of our application through a web browser, following the HTTP (Hypertext Transfer Protocol) protocol. We chose this transfer protocol, because it's probably the most used protocol for the communication of information on the web, especially in a client-server application as the one we decided to adopt.

Almost all the components we identified are installed on the application server, so they are located on the same machine in order to facilitate the direct communication between them.

As for the interface between the application server and the database server, we chose a simple implementation: for now, these two servers are in the same location, so they are connected through a local network. The information are retrieved from the database simply using SQL queries.

We're considering for the future a better and more secure implementation, using a remote database, accessed through the SQL*Net networking software, and the cloud for the back-up.

2.7 Selected architectural styles and patterns

2.7.1 Architectural style

We selected a single architectural style for our application: a **Client-Server 3-tier architecture**.

This means that there are at least two different processes, which are located on different hosts. The two processes have well-defined interfaces that allow the access and each process is characterized by different roles:

- ***Client***: the client is the process that plays an active role, working to make service requests directed to the server. It's the client that initializes the communication, by sending messages and throwing remote calls. In our case, the client is a "thin client", which depends heavily on the server.
- ***Server***: the server, usually plays a passive role in a client-server architecture. It simply receives requests and provides answers. It's therefore able to offer a certain set of services. In our case, we have two separated servers situated in the same local network: the application server and the database server.

So, this is the architecture style we chose to design and define the architecture of our system: we decided to use this kind of architectural style because of the benefits that it can offer to our application, especially in terms of performance and availability.

Moreover, thanks to this solution we can guarantee an improved data integrity, because all the data passes in the middle tier in order to be stored and retrieved, so data corruption through client applications can be eliminated (clients can not directly access to the data). This also allows to enhance the security of our system.

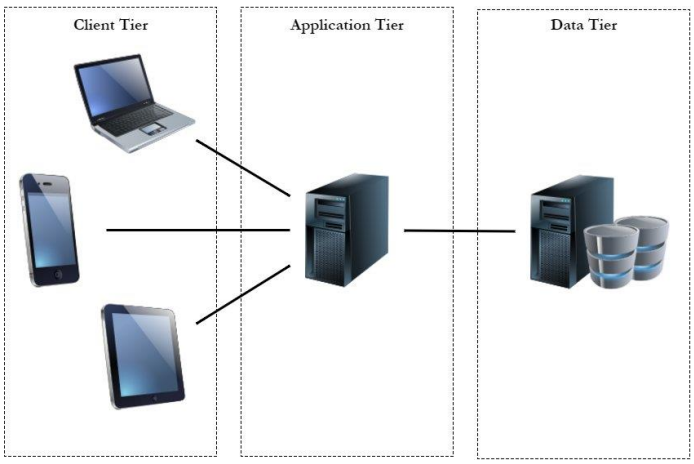
Our 3-tier architecture is quite similar to the standard one: a first tier web-browser-based graphical user interface, usually at a personal computer, a middle-tier logic application, situated on a local network and a third tier for the data. In particular, we have a Database Server where two relational

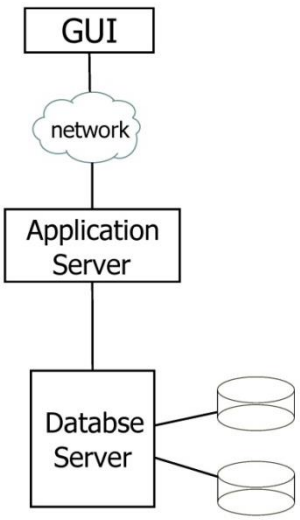
databases are located on two different machines and managed by two different sub-systems. In this way, we have the main database where all the relevant data and information for the application are stored and another one which is used for the back-up. The system, automatically, execute a back-up and updates the copy of the main database daily, when the service offered by our application is not available (between 9:00 p.m. and 6:00 a.m.). On this Database Server we have decided to use a RAID composed by two disks, one for each relational database. In this way we can manage the data back-up, implementing the RAID 1 level (mirroring). This means that we simply create a redundant copy of the main disk on the mirroring disk. In this way, all the data contained in the first disk are copied identically on the second one.
We decided to adopt this kind of implementation in order to increase the reliability and the fault tolerance. For now, this seemed the simplest and the most suitable choice, but maybe in the future we will consider switching to a RAID 5 level implementation.

After a general description of our architecture, we now describe more accurately the physical levels that characterized our system:

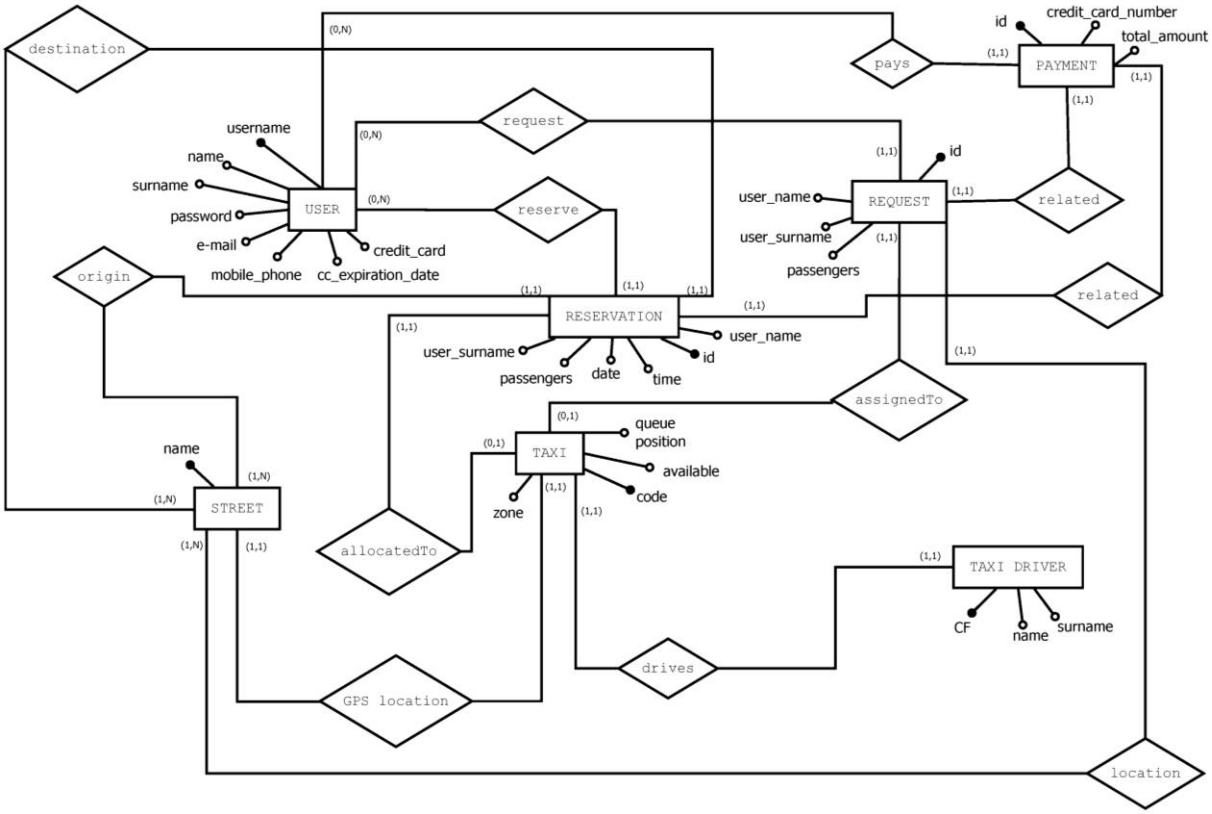
1. **Client Tier**: this tier is the top level of the application and it represents the user interface. The main function of the interface is to translate tasks and results to something the user can understand. Here our client is thought to be a “thin client” that only contains the presentation logic. In simple terms it is a layer which users can access directly, such as a web page, like in our case.
2. **Application Tier**: this tier contains the logic of the application, it coordinates all the actions and the whole application. It controls all the functionalities and performs processes. It receives all the user commands, interprets them and executes all the related operations. This is the core of the application, the business logic that coordinates everything and allows data access.
3. **Data Tier**: this tier is where all the data and information are stored and retrieved. The data tier includes the data persistence mechanisms and the data access layer, that exposes methods for managing the stored data.

Each one of these tiers are divided into three different levels: Client (Client Tier), Application Server (Application Tier) and Database Server (Data Tier).





Finally, we want to provide a simple ER diagram in order to illustrate which data are store in our relational database:



2.7.2 Architectural pattern: MVC

This pattern allows the separation between data model, application logic and data presentation. In fact, these elements are implemented in three distinct components:

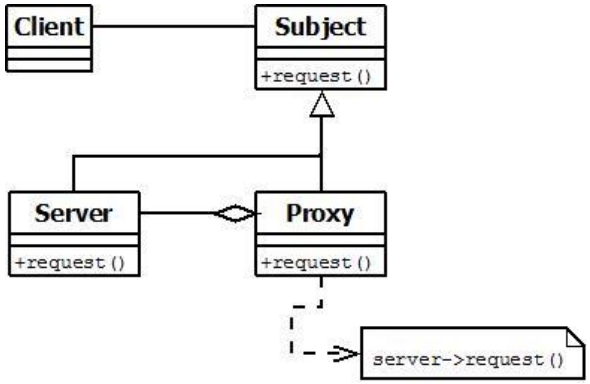
- ***Model***: representation of the information concerning the specific application domain. It's an abstract representation of our system, simply including data and methods used to process them.
- ***View***: it presents data in such a way that it is possible interacting with them, typically through a user interface. In our case it lies in the interaction between users and application, and between taxi drivers and application.
- ***Controller***: it receives inputs from the view, converting them into actions that the model can execute and it responds to events calling for changes to the model and sometimes also on the view. It corresponds to the subsystems that we found: Sign Up System, Registered User Manager, Taxi Manager, Payments Manager, Data System.

2.7.3 Design patterns

Creational pattern: Proxy

The “Proxy” pattern provides a representation of an object difficult to access or which requires an important time for the access or for its creation. In particular, we adopted an “ad hoc” solution for our system, using the “Proxy remote” pattern that provides a reference to an object located in a different address space, in our case in the database. This pattern is useful when the system has to send a notification to a user and therefore it must retrieve his e-mail address stored in the database. Similarly, during the payments control, the system uses this kind of pattern to retrieve from the database the payment history of the selected user.
So, we decide to use a proxy in order to facilitate the access to the database.

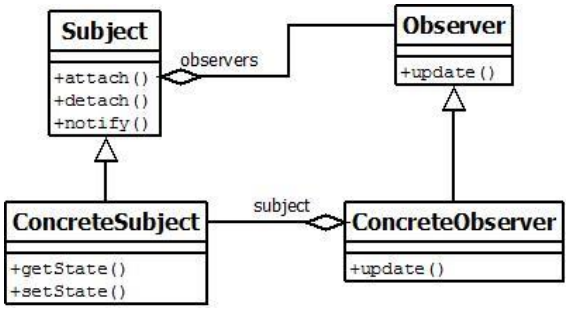
Below, a graphical UML representation of the pattern:



Behavioral pattern: Observer

This pattern defines a one-to-many dependency between different objects, so that if an object changes its state, all the interested objects can be notified of the change and eventually updated.
In our case, the subject (the observed object) is the system, while the taxi drivers can be considered as observers (or listeners).
Therefore, the system notifies the observers about changes in its current state, such us activity / inactivity of the platform, down service or traffic notification, which represent extraordinary notifications sent to all the taxi drivers (broadcast communication).

Below, a graphical UML representation of the pattern:



3 Algorithm Design

In order to execute specific operations and accomplish their tasks, the managers of the application exploit dedicated algorithms.

We decided to focus on two of the most important algorithms that are part of our project: the first algorithm has the objective of managing the taxi queues in each zone of the city, while the second one is the well-known Dijkstra algorithm, used to compute the shortest path for a taxi in order to reach the location where the passenger required the taxi. This shortest path is then communicated to the taxi driver on his on board mobile app.

Management of taxis queues:

We use the concept of "queue" in the standard definition. A queue is a data structure based on a FIFO ("First In First Out") mechanism: this means that the first element inserted in the queue is the first one that goes out. For our application, we imagine that the elements of the queues are all the taxi of the city, each one identified by its unique code. In the application database, each taxi is also associated with a number, that obviously can change, representing its position in the queue and a letter identifying the zone in which the taxi is located.

There are different queues, one for each zone of the city, and each zone is identified by a different name (Zone A, Zone B, Zone C and Zone D).

When a request arrives, the first taxi in the queue of the requested zone is notified: if it's occupied, it is moved to the bottom of the queue and the request is forwarded to the following taxi; otherwise, if the taxi is available, it accepts the request and it's also moved to the bottom of the queue.

So, as for the algorithm, we have a function **enqueue(Q,x)** that inserts the element *x* in the queue, in the last position. This function is useful at the beginning of the day, when the application starts and all the queues are created or when a taxi needs to be moved to the bottom of the same queue of another queue (this can happen when a taxi ends its ride in a different zone with respect to the starting zone).

The function **dequeue(x)** removes the first element *x* (the first taxi) and returns it.

The function **isEmpty(Q)** returns a boolean specifying whether a queue is empty: considering our assumption ("There must be always a taxi in every queue") listed in the RASD document, this function will always return false.

The function **front(Q)** returns the first element of the queue, that is, in our case, the first taxi in the queue. We also define two important attributes that can help in the execution of the operations on the different queues: ***Q.head*** represents the beginning of the queue and ***Q.tail*** indicates the position of the next element (where a new element will be inserted in queue). So, the elements of the queue are in the positions *Q.head, Q.head+1,…, Q.tail-1*. The queue is managed according to a circular orderd: this means that if we have a queue with *n-1* elements (index from1 to n), after position *n*, we have position *1*. When *Q.head = Q.tail* the queue is empty, while if *Q.head = Q.tail = 1* the queue is full.

Here a simple pseudocode of the algorithm we have just described:

```
ENQUEUE(Q,x)
1  Q[Q.tail] = x
2   if Q.tail == Q.length        // if we are at the end of the array
3     Q.tail = 1                 // we start again from the first position
4   else  Q.tail = Q.tail + 1

DEQUEUE(x)
1   x = Q[Q.head]
2   if Q.head == Q.length        // if we are at the end of the array
3     Q.head = 1                 // we start again from the first position
4   else  Q.head = Q.head + 1

IsEMPTY(Q)
1  if Q.head == Q.tail
2     return true
3  else  Q.tail = Q.tail + 1

FRONT(Q)
1  return Q[Q.head]
```

Shortest path for a taxi:

In order to find the shortest path between two locations, we decided to use a well-known algorithm: Dijsktra's algorithm. We try to reduce the real-world situation to a simple directed graph, in order to execute the algorithm: in our case, each node of the graph represents a street (identified by a progressive number according to alphabetical order) and each arc represents a path connecting two nodes. The cost of each arc is the time (expressed in minutes) needed to go from a node to the other.

Here the definition of the algorithm and the pseudocode describing all the steps:

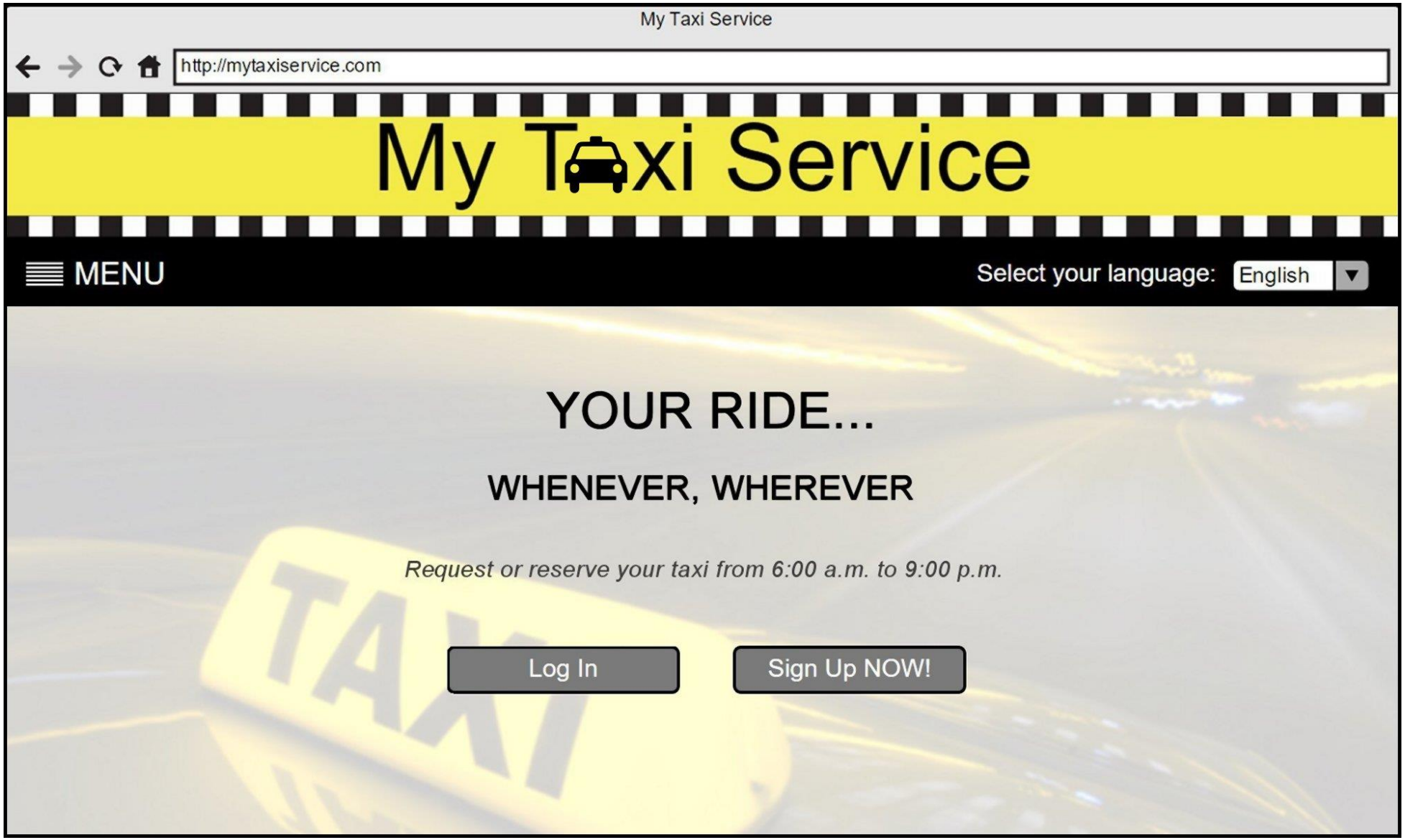
→ Given a directed graph *G(N,A)*, with *n* nodes and *m* arcs, a cost c_{ij} for each arc *(i,j)* connecting node *i* to node *j* (with $c_{ij} = +\infty$, if $(i,j) \notin A$) and a node $s \in N$, the Dijkstra's algorithm allows us to find the shortest paths from *s* to all the other nodes of the graph.

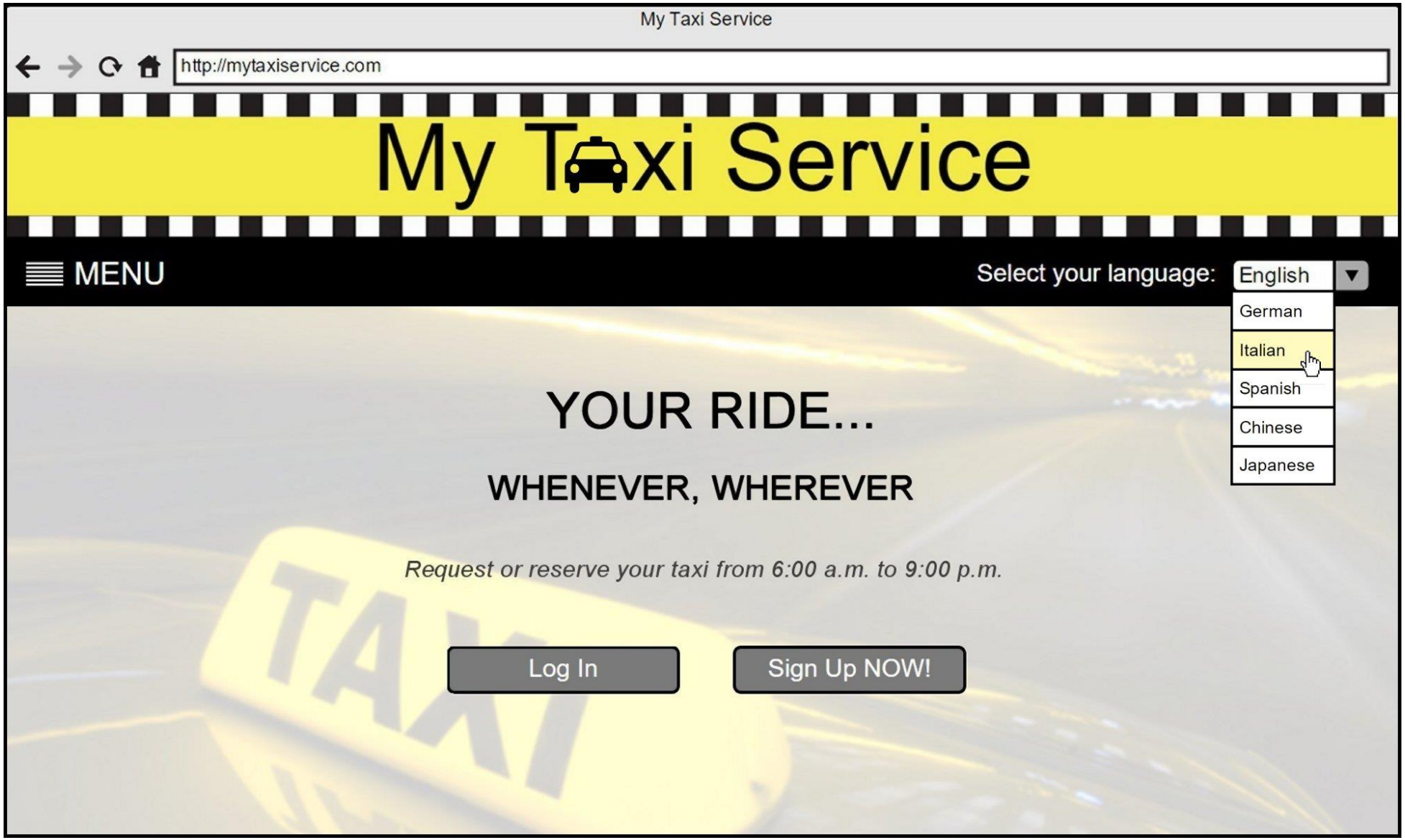
```
BEGIN
      S := {s};            // S = subset of nodes already considered
      L[s] := 0;           // L[j] = cost of a shortest path from s to j
      pred[s] := s;        // pred[j] = predecessor of j in the shortest path from s to j

      WHILE |S| ≠ n DO
      select (v,h) ∈ δ+(S) = {(i,j) : (i,j)∈A, i∈S, j∉S} such that
            L[v] + cvh = min {L[i] + cij : (i,j) ∈ δ+(S)};
      L[h] := L[v] + cvh;
      pred[h] := v;
      S := S ∪ {h};
      END-WHILE
END
```

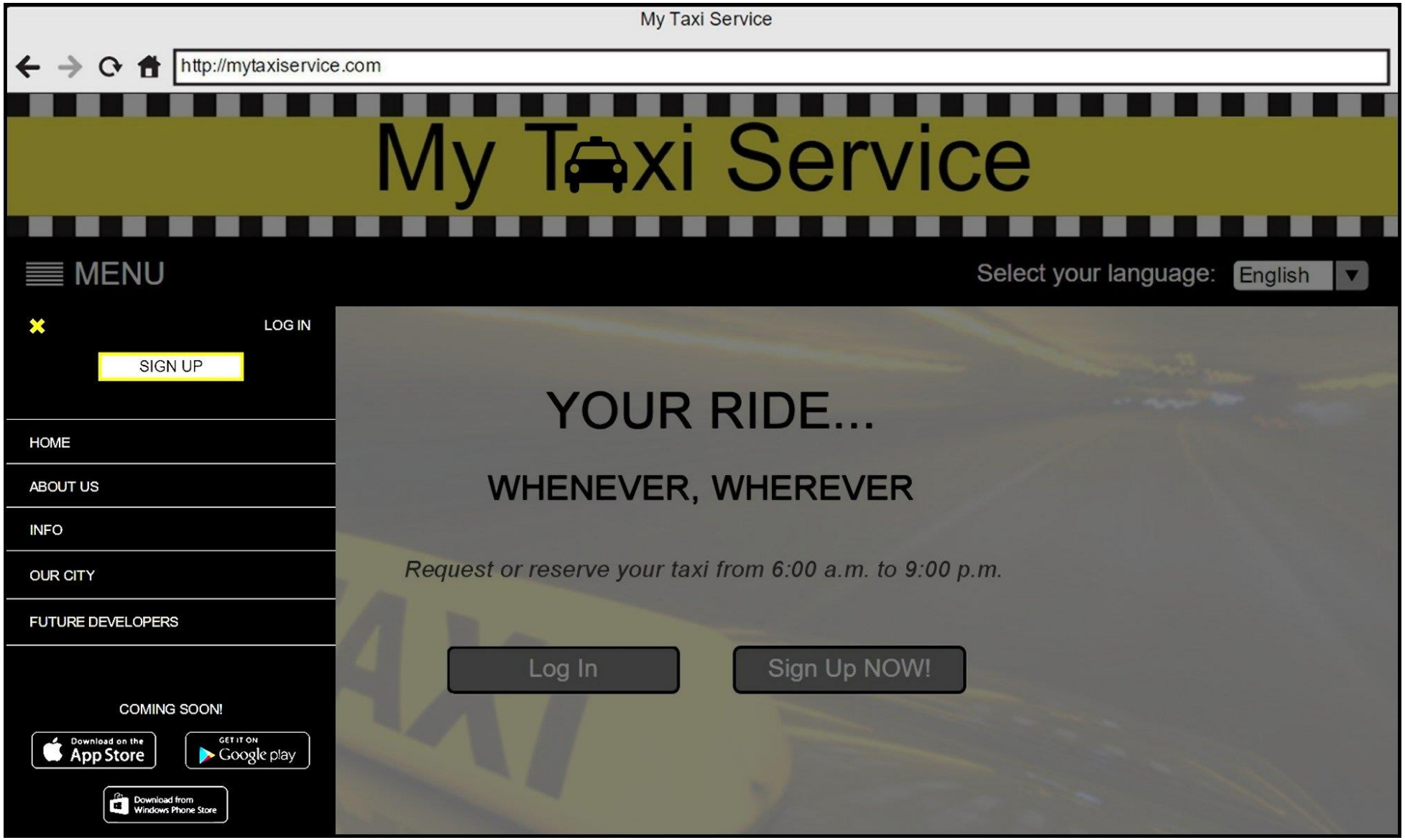
4 User Interface Design

We provide in the following pages some mockups representing how the user interfaces of our system will look like.
In the RASD document we have already provided few mockups representing our idea of the structure of the most important application pages. However, those mockups represented our initial idea of the interfaces and were very essentials and schematic.
We now want to show how our application will really be and how all the functionalities offered can be exploited. We represent the web application interface for the guests and registered users and the on board mobile app of each taxi driver.
In a near future, a mobile application for each kind of smartphone (Android, IOS, Windows Phone) is planned to be developed and implemented, but for now the service is offered only through the web and can be exploited (on PCs, smartphones and tablets) using any web browser and an internet connection.

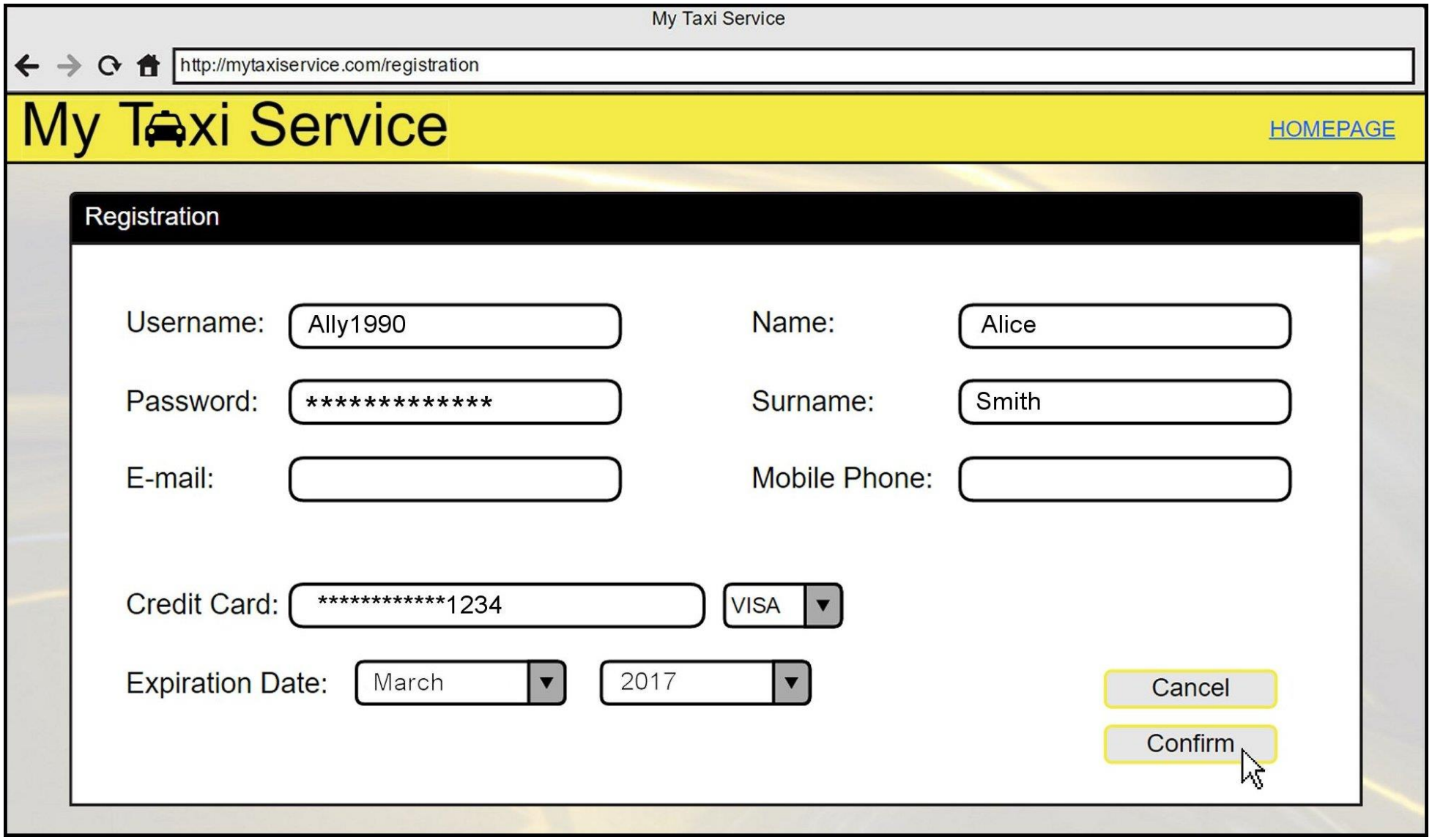
The following mockups are thought to be browsed sequentially, in order to correctly visualize the flow of the navigation and the consequences of each action or operation.
Obviously, not all the possible functionalities and pages are show, because honestly this would have been too long; however, we have drawn a remarkable number of mockups showing almost all the functionalities offered.

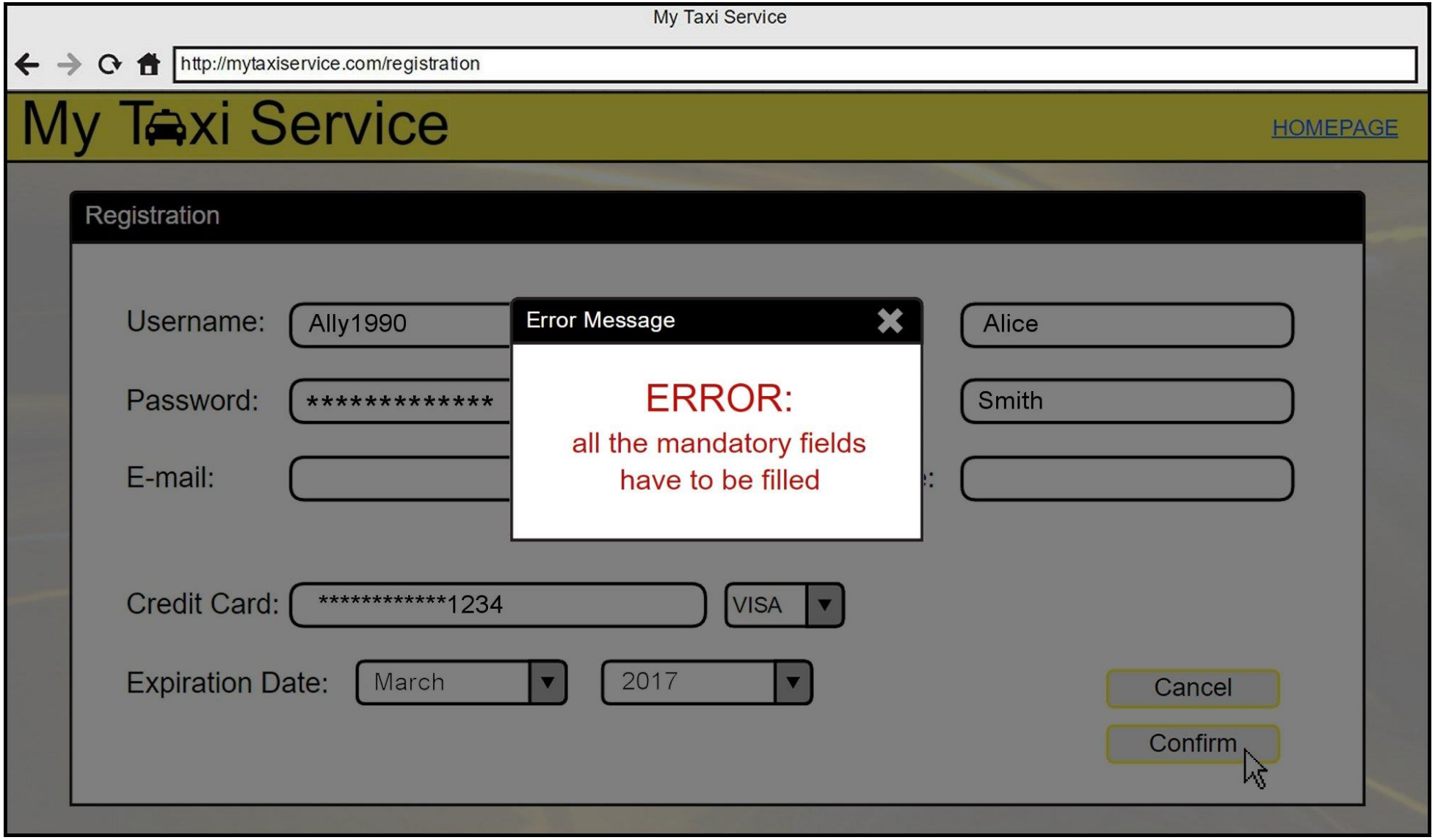

My Taxi Service
http://mytaxiservice.com
My Taxi Service
MENU
Select your language:
English
YOUR RIDE...
WHENEVER, WHEREVER
Request or reserve your taxi from 6:00 a.m. to 9:00 p.m.
Log In
Sign Up NOW!
TAXI

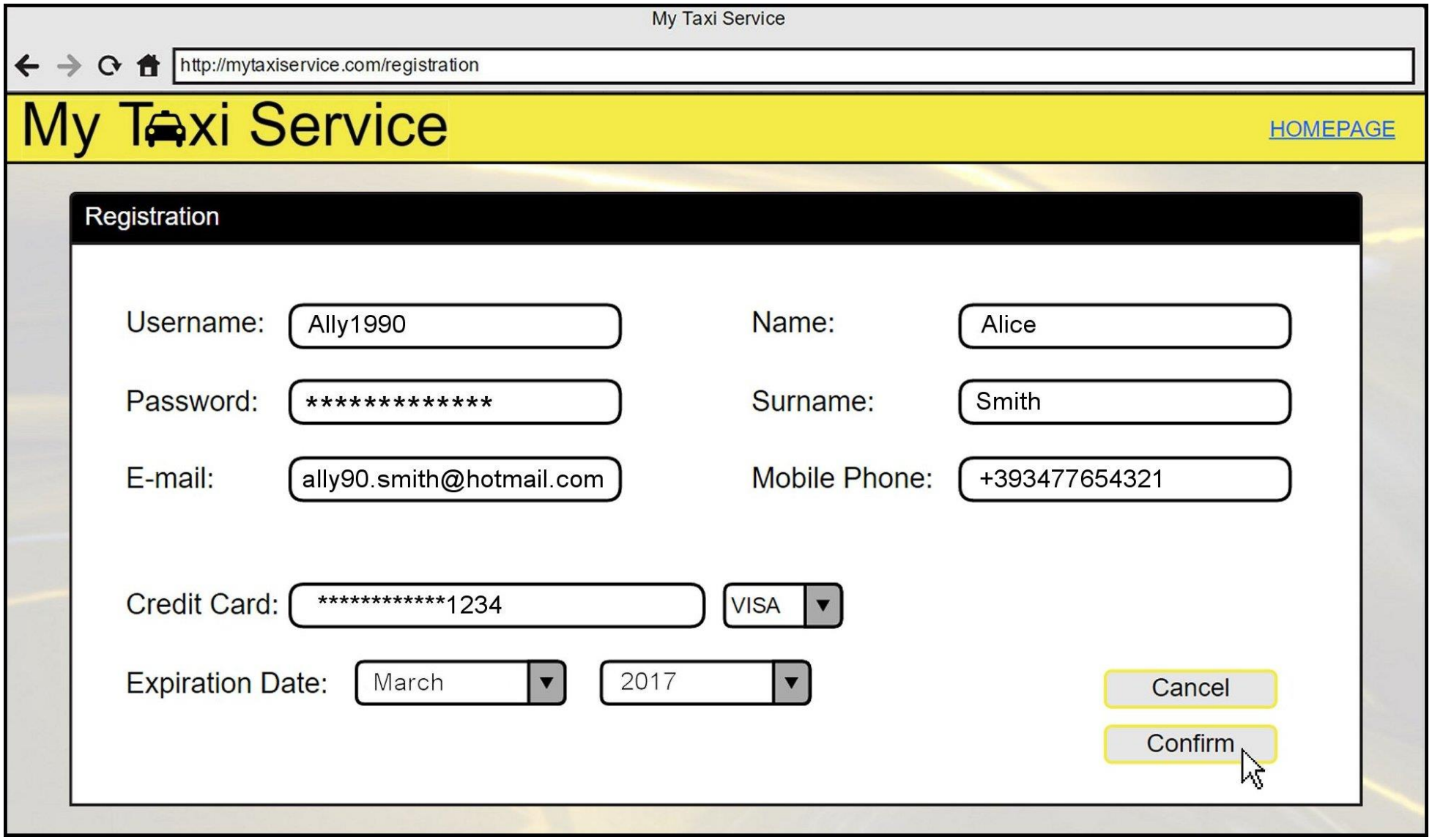

My Taxi Service
http://mytaxiservice.com
My Taxi Service
MENU
Select your language:
English
German
Italian
Spanish
Chinese
Japanese
YOUR RIDE...
WHENEVER, WHEREVER
Request or reserve your taxi from 6:00 a.m. to 9:00 p.m.
Log In
Sign Up NOW!
TAXI

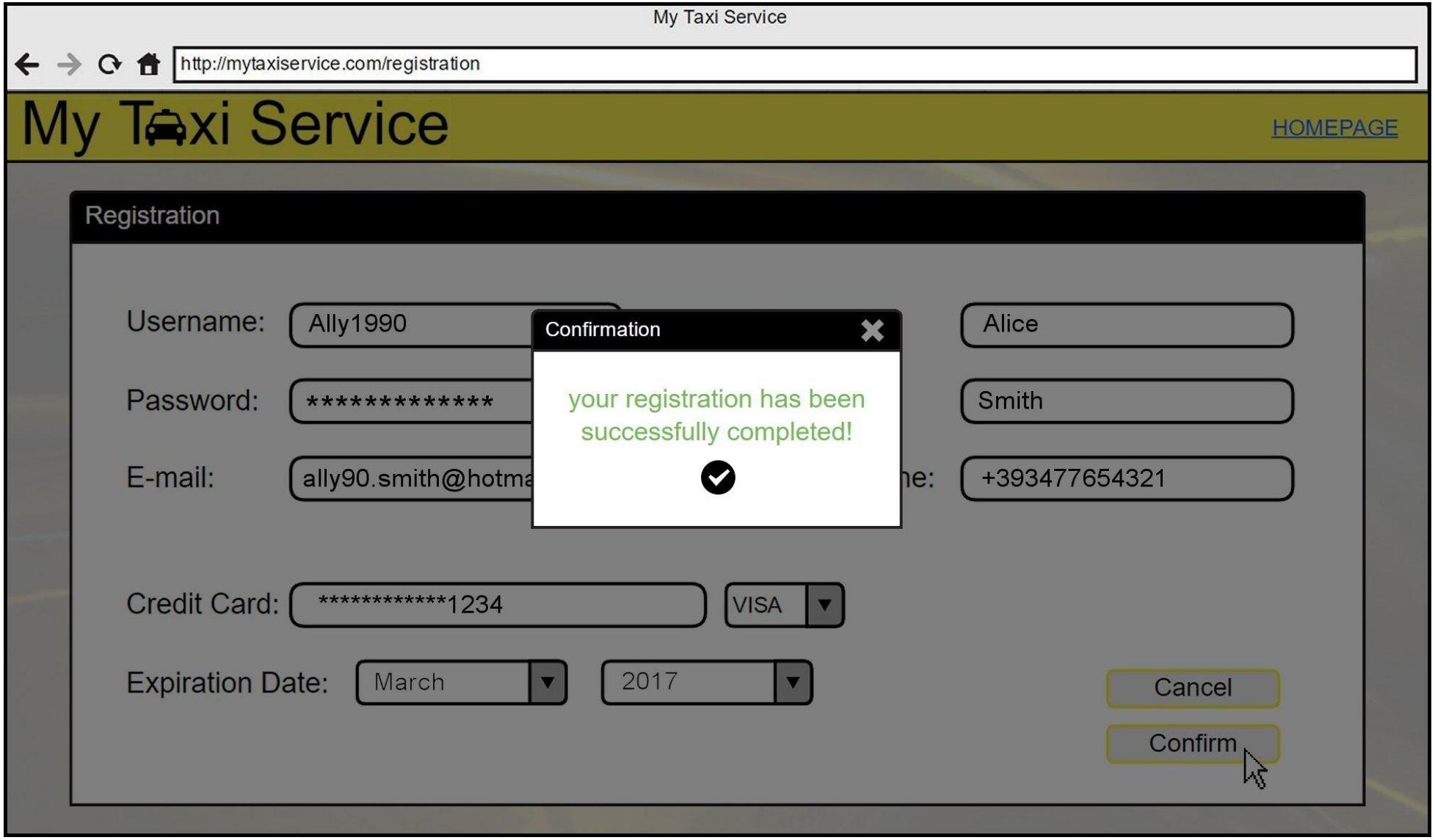

My Taxi Service
http://mytaxiservice.com
My Taxi Service
MENU
Select your language:
English
LOG IN
SIGN UP
HOME
ABOUT US
INFO
OUR CITY
FUTURE DEVELOPERS
COMING SOON!
Download on the App Store
GET IT ON Google play
Download from Windows Phone Store
YOUR RIDE...
WHENEVER, WHEREVER
Request or reserve your taxi from 6:00 a.m. to 9:00 p.m.
Log In
Sign Up NOW!

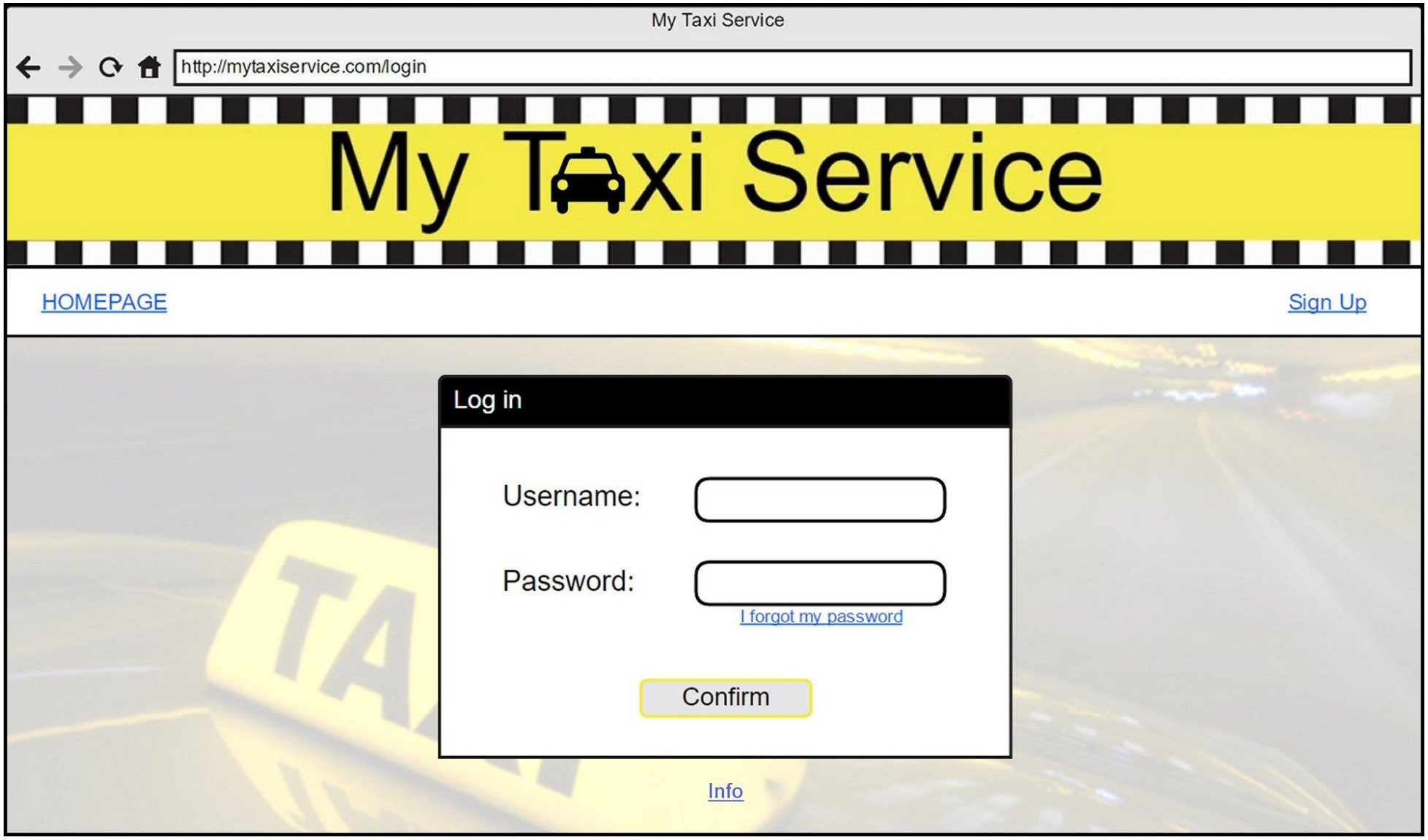

My Taxi Service
http://mytaxiservice.com/registration
My Taxi Service
HOMEPAGE
Registration
Username:
Password:
E-mail:
Name:
Surname:
Mobile Phone:
Credit Card:
VISA
Expiration Date:
select a month
select a year
Cancel
Confirm

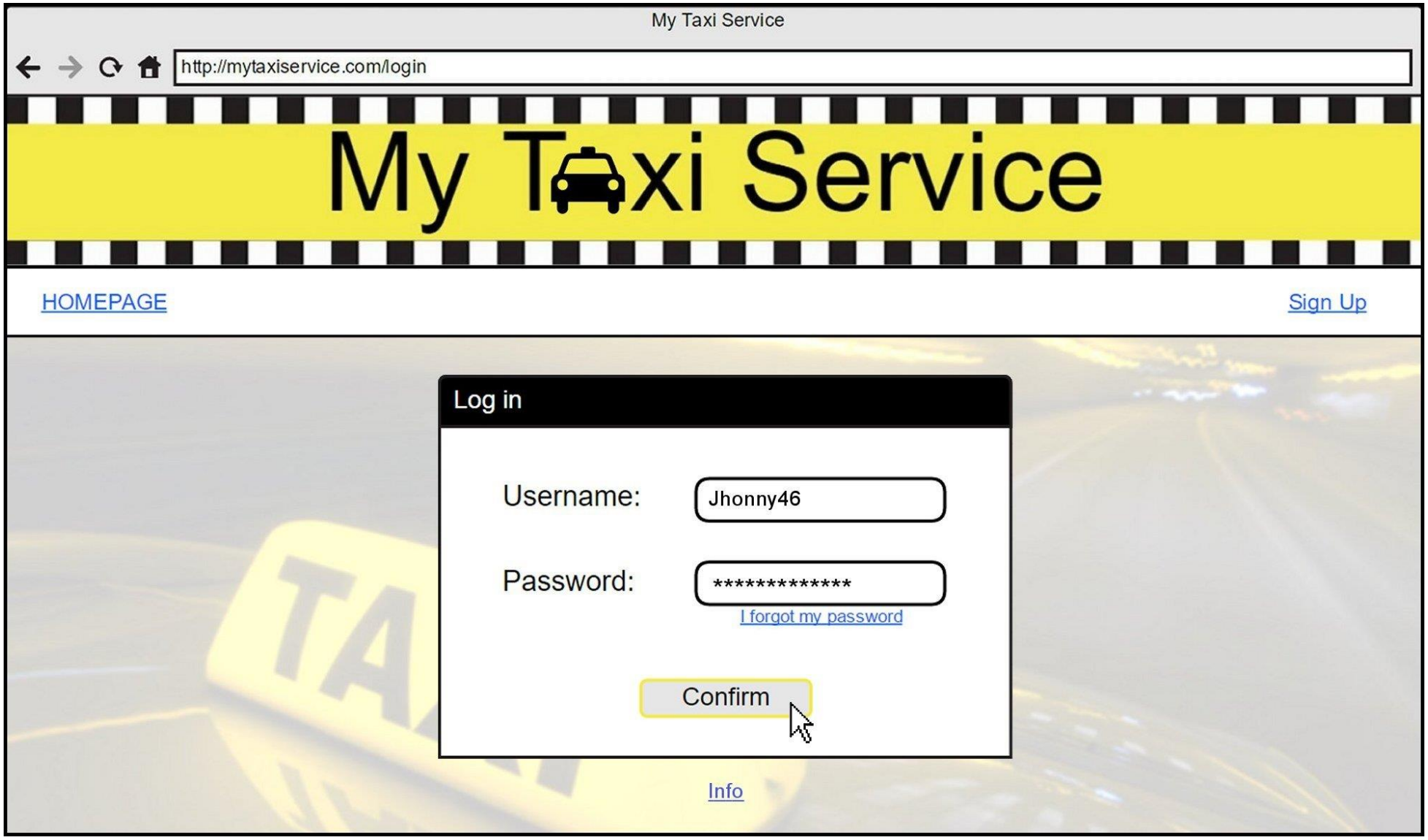

My Taxi Service
http://mytaxiservice.com/registration
My Taxi Service
HOMEPAGE
Registration
Username:
Ally1990
Name:
Alice
Password:
****
Surname:
Smith
E-mail:
Mobile Phone:
Credit Card:
**********1234
VISA
Expiration Date:
March
2017
Cancel
Confirm

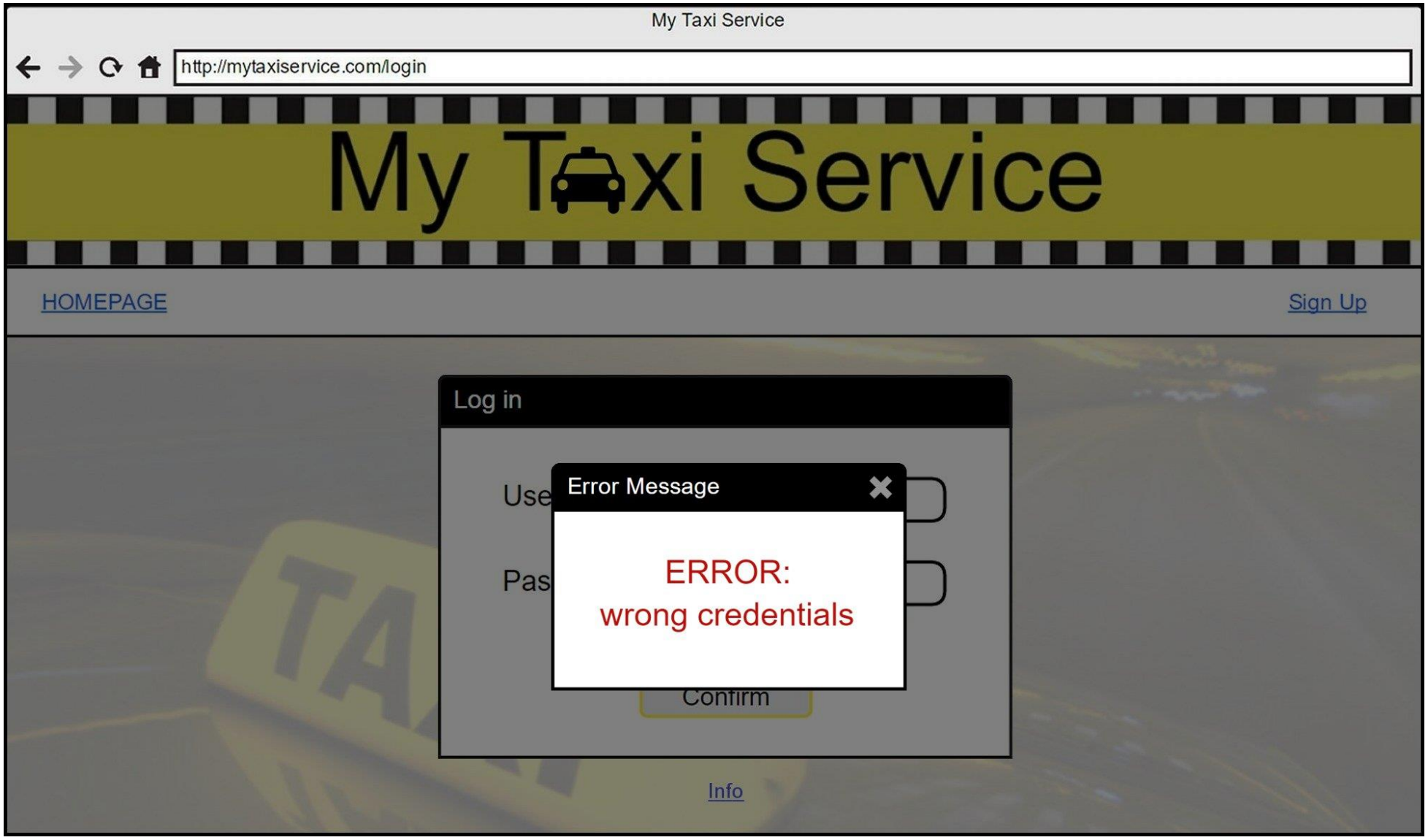

My Taxi Service
http://mytaxiservice.com/registration
My Taxi Service
HOMEPAGE
Registration
Username:
Ally1990
Password:
****
E-mail:
Error Message
ERROR:
all the mandatory fields
have to be filled
Alice
Smith
Credit Card:
**********1234
VISA
Expiration Date:
March
2017
Cancel
Confirm

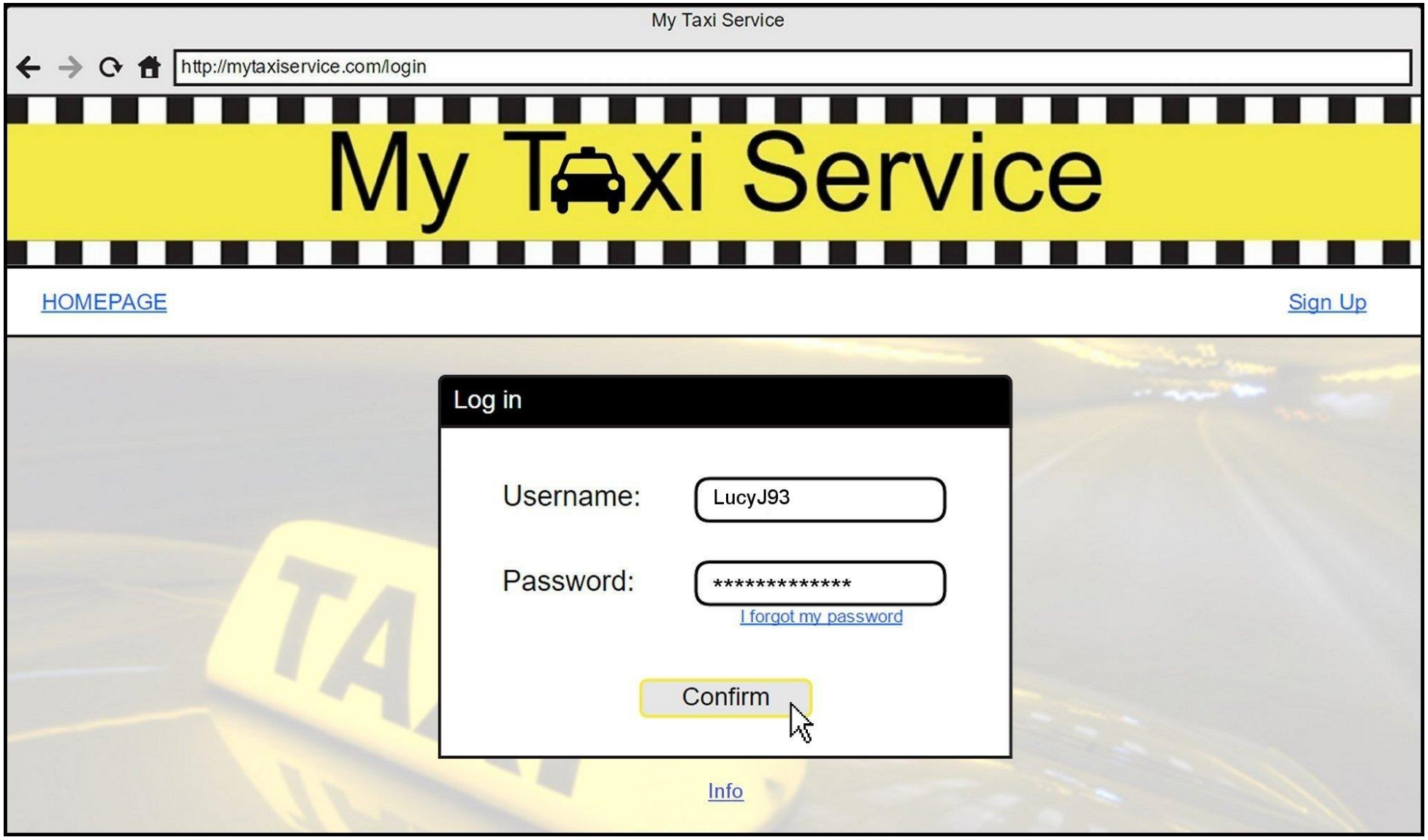

My Taxi Service
http://mytaxiservice.com/registration
My Taxi Service
HOMEPAGE
Registration
Username:
Ally1990
Name:
Alice
Password:
****
Surname:
Smith
E-mail:
ally90.smith@hotmail.com
Mobile Phone:
+393477654321
Credit Card:
**********1234
VISA
Expiration Date:
March
2017
Cancel
Confirm


My Taxi Service
http://mytaxiservice.com/registration
My Taxi Service
HOMEPAGE
Registration
Username:
Ally1990
Password:
Alice
Smith
E-mail:
+393477654321
Credit Card:
**********1234
VISA
Expiration Date:
March
2017
Cancel
Confirm
Confirmation
your registration has been successfully completed!


My Taxi Service
http://mytaxiservice.com/login
My Taxi Service
HOMEPAGE
Sign Up
Log in
Username:
Password:
I forgot my password
Confirm
Info


My Taxi Service
http://mytaxiservice.com/login
My Taxi Service
HOMEPAGE
Sign Up
Log in
Username:
Jhonny46
Password:
****
I forgot my password
Confirm
Info


My Taxi Service
http://mytaxiservice.com/login
My Taxi Service
HOMEPAGE
Sign Up
Log in
Use
Pas
Error Message
ERROR:
wrong credentials
Confirm
Info


My Taxi Service
http://mytaxiservice.com/login
My Taxi Service
HOMEPAGE
Sign Up
Log in
Username:
LucyJ93
Password:
****
I forgot my password
Confirm
Info

My Taxi Service

http://mytaxiservice.com/lucy-johnson

My Taxi Service

Lucy | Personal Info | Your Reservations | Logout

Lucy Johnson

Hi Lucy,
what do you want to do?

REQUEST

RESERVE

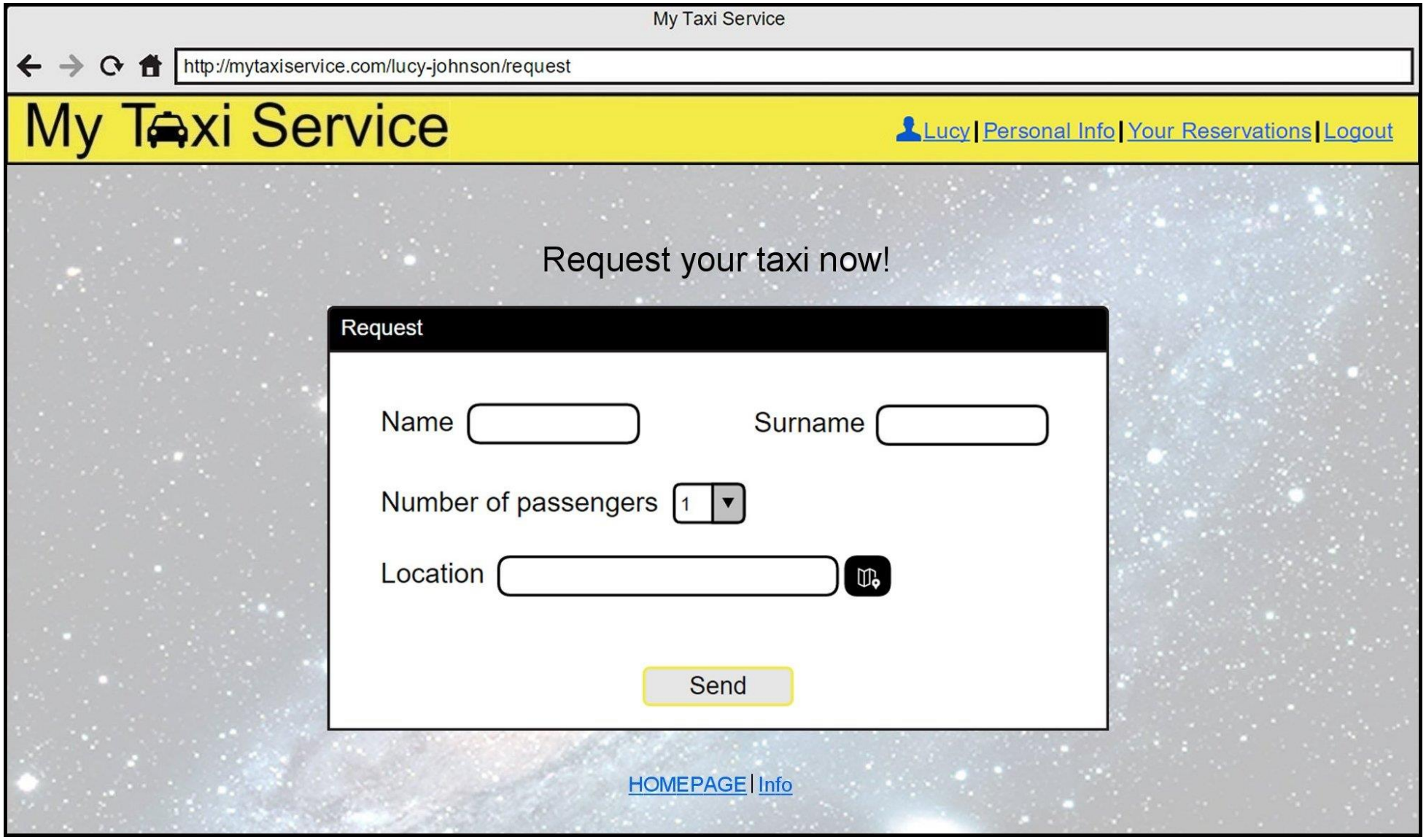
change your profile picture

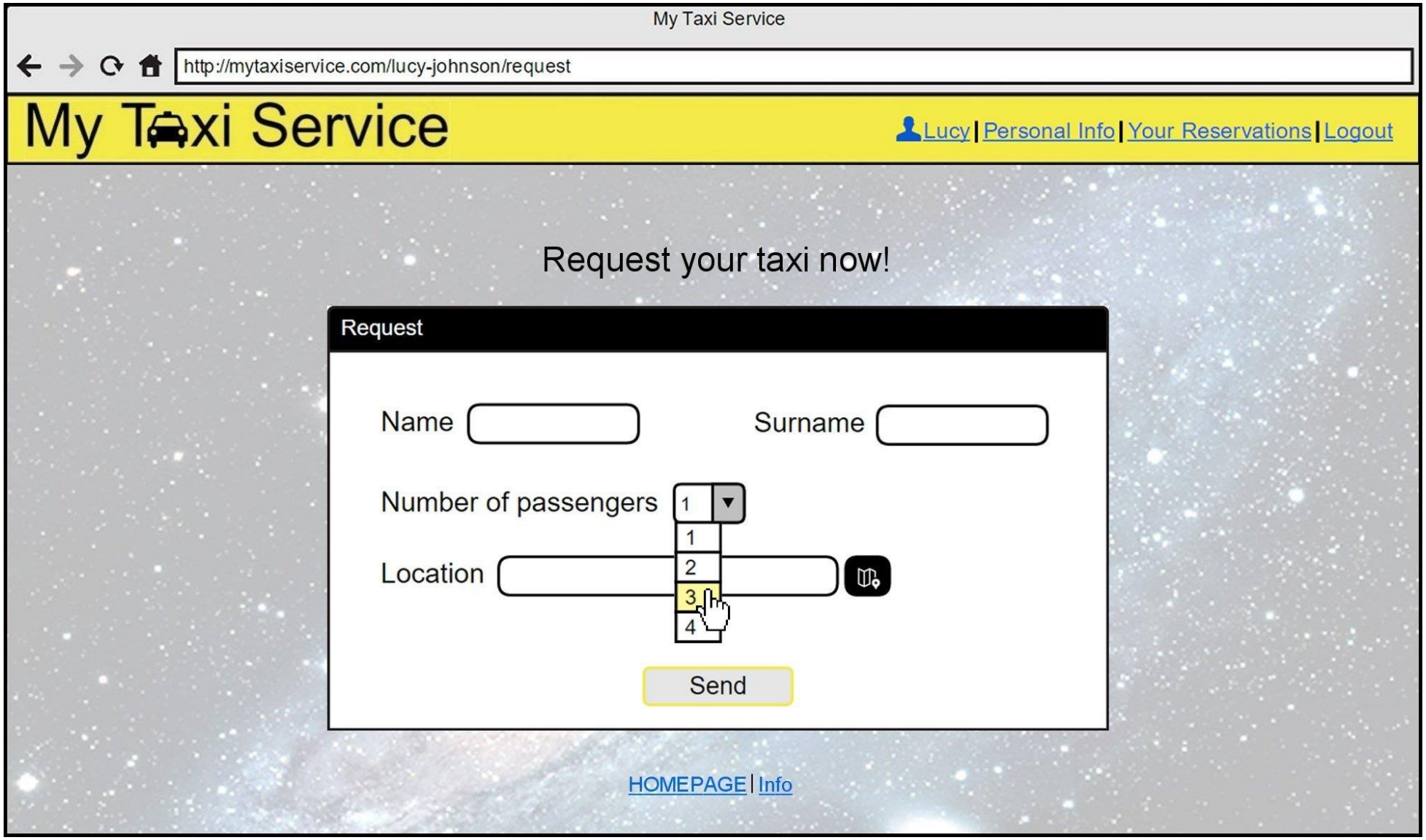
change your background image

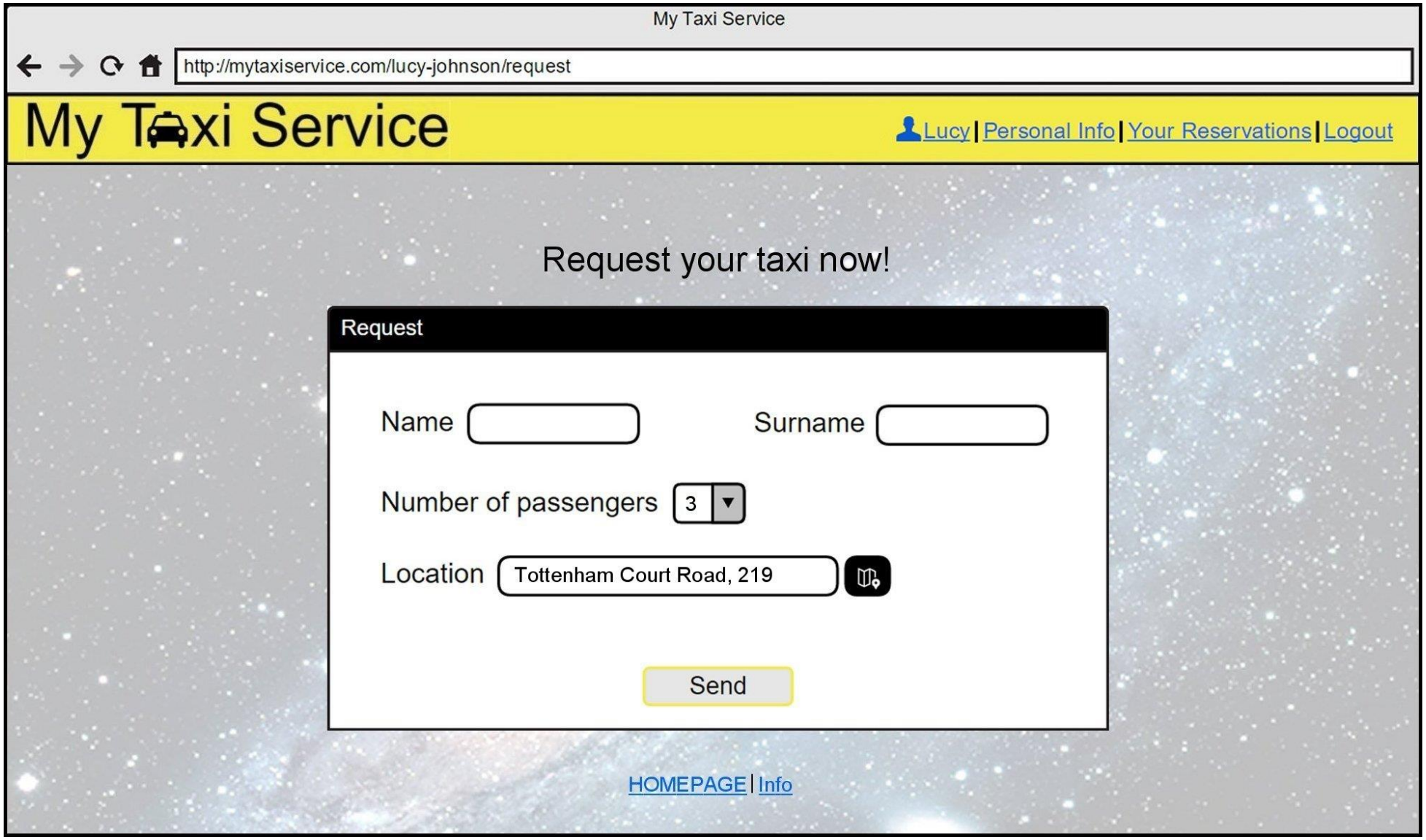
Notifications

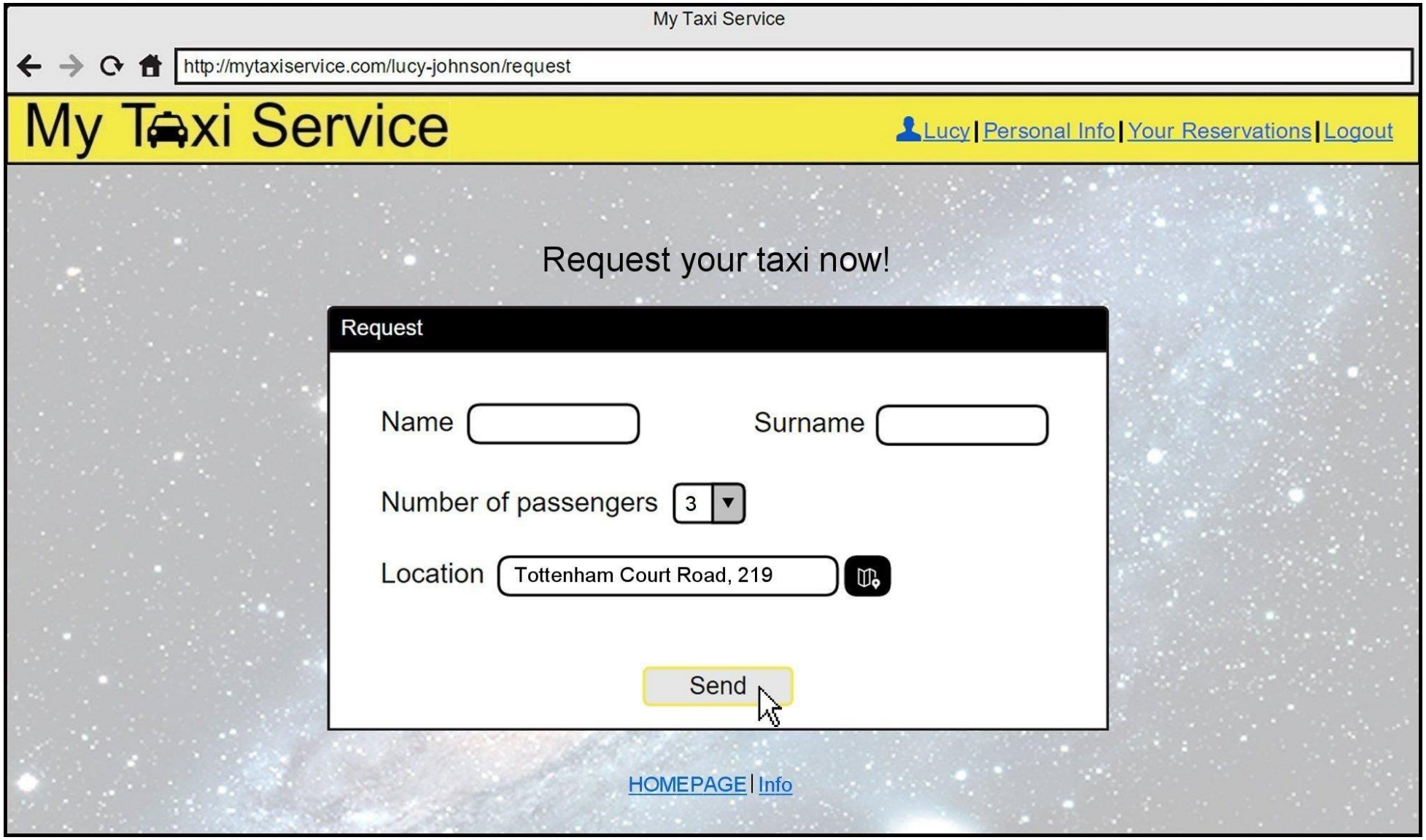
Your reservation for December, 10th has been successfully registered

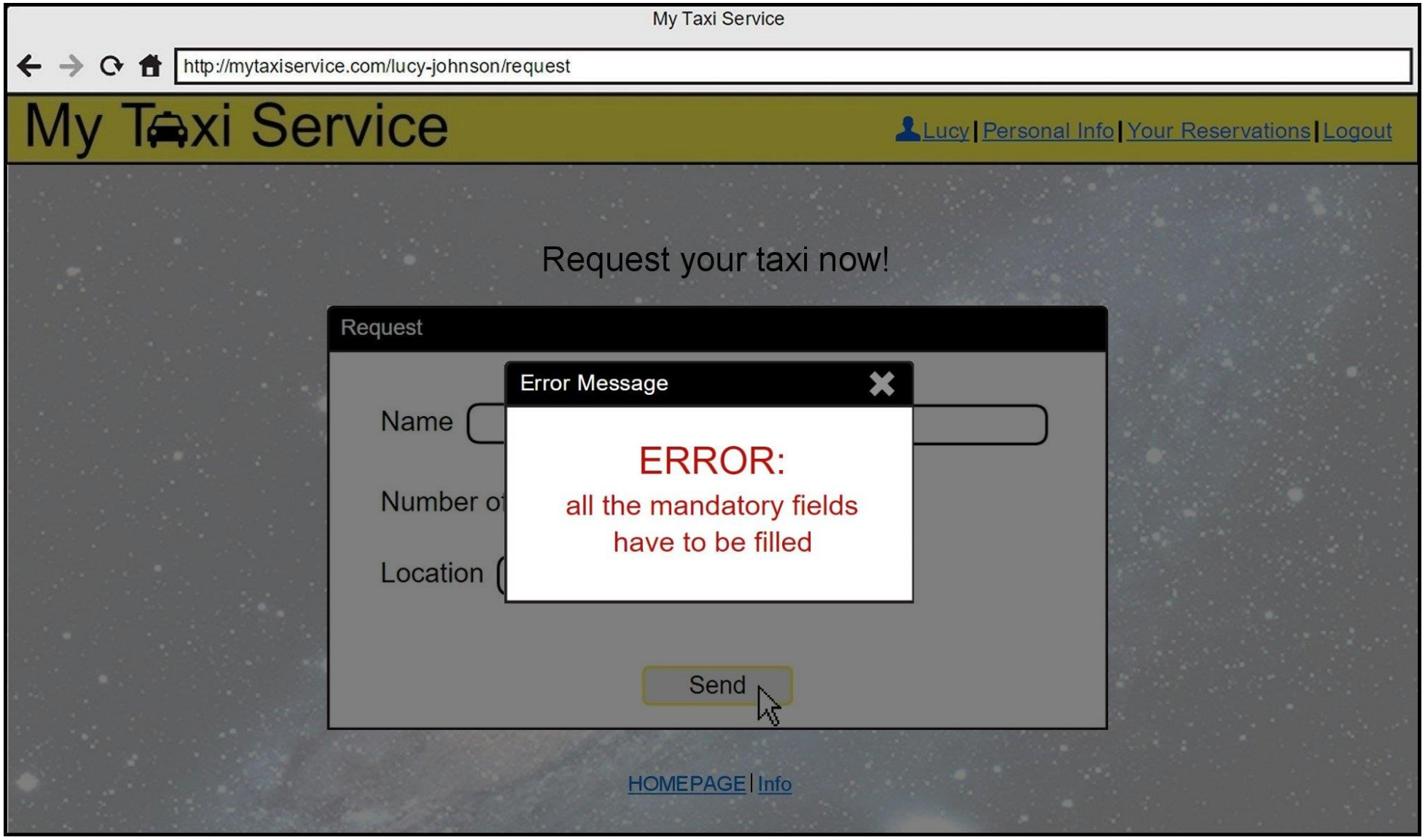
HOMEPAGE | Info

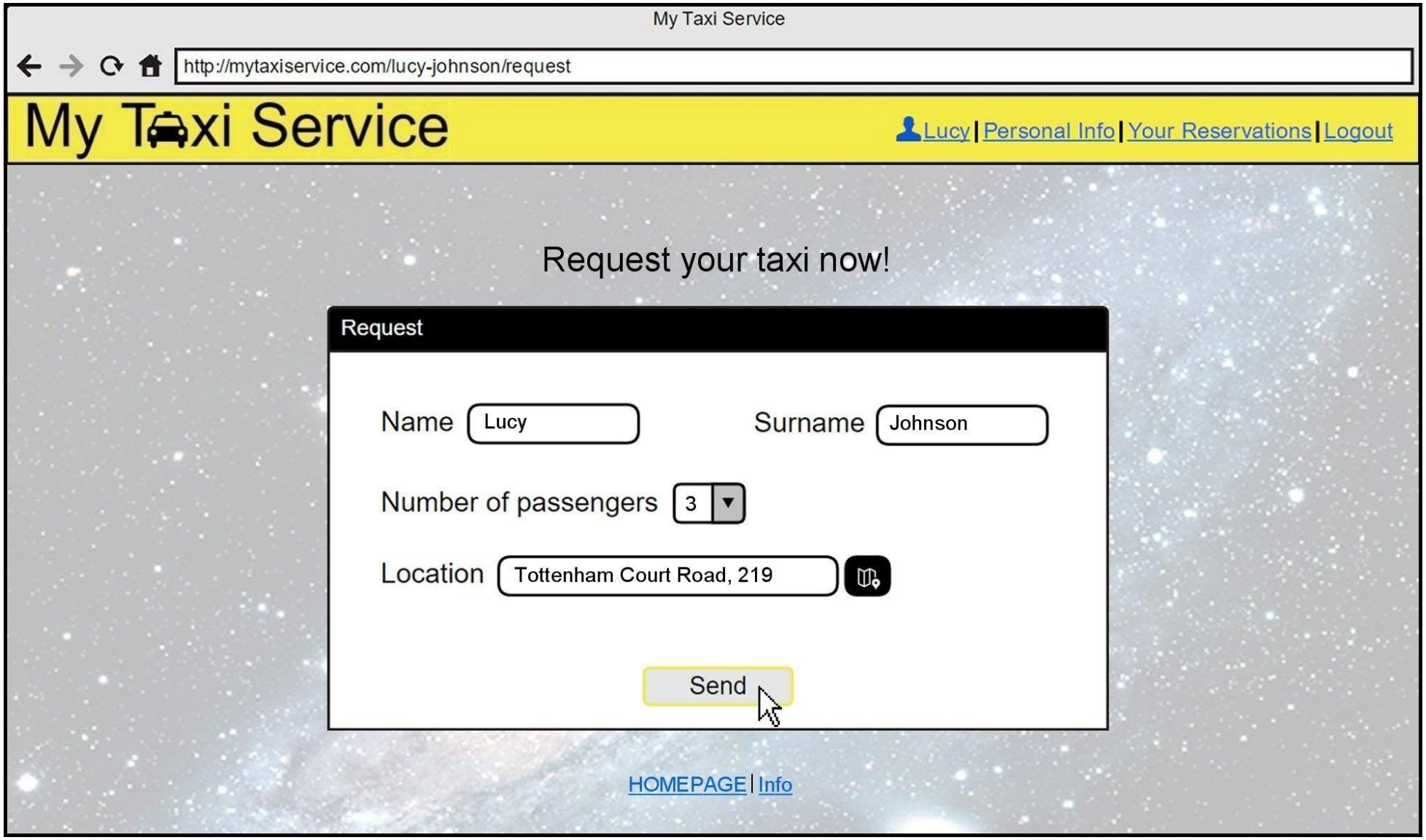

My Taxi Service
http://mytaxiservice.com/lucy-johnson/request
My Taxi Service
Lucy | Personal Info | Your Reservations | Logout
Request your taxi now!
Request
Name
Surname
Number of passengers
1
Location
Send
HOMEPAGE | Info


My Taxi Service
http://mytaxiservice.com/lucy-johnson/request
My Taxi Service
Lucy | Personal Info | Your Reservations | Logout
Request your taxi now!
Request
Name
Surname
Number of passengers
1
1
2
3
4
Location
Send
HOMEPAGE | Info


My Taxi Service
http://mytaxiservice.com/lucy-johnson/request
My Taxi Service
Lucy | Personal Info | Your Reservations | Logout
Request your taxi now!
Request
Name
Surname
Number of passengers
3
Location
Tottenham Court Road, 219
Send
HOMEPAGE | Info


My Taxi Service
http://mytaxiservice.com/lucy-johnson/request
My Taxi Service
Lucy | Personal Info | Your Reservations | Logout
Request your taxi now!
Request
Name
Surname
Number of passengers
3
Location
Tottenham Court Road, 219
Send
HOMEPAGE | Info


My Taxi Service
http://mytaxiservice.com/lucy-johnson/request
My Taxi Service
Lucy | Personal Info | Your Reservations | Logout
Request your taxi now!
Request
Error Message
Name
ERROR:
all the mandatory fields
have to be filled
Number of
Location
Send
HOMEPAGE | Info


My Taxi Service
http://mytaxiservice.com/lucy-johnson/request
My Taxi Service
Lucy | Personal Info | Your Reservations | Logout
Request your taxi now!
Request
Name
Lucy
Surname
Johnson
Number of passengers
3
Location
Tottenham Court Road, 219
Send
HOMEPAGE | Info

My Taxi Service

http://mytaxiservice.com/lucy-johnson

My Taxi Service

Lucy | Personal Info | Your Reservations | Logout

Lucy Johnson

Hi Lucy,

what do you want to do?

REQUEST

RESERVE

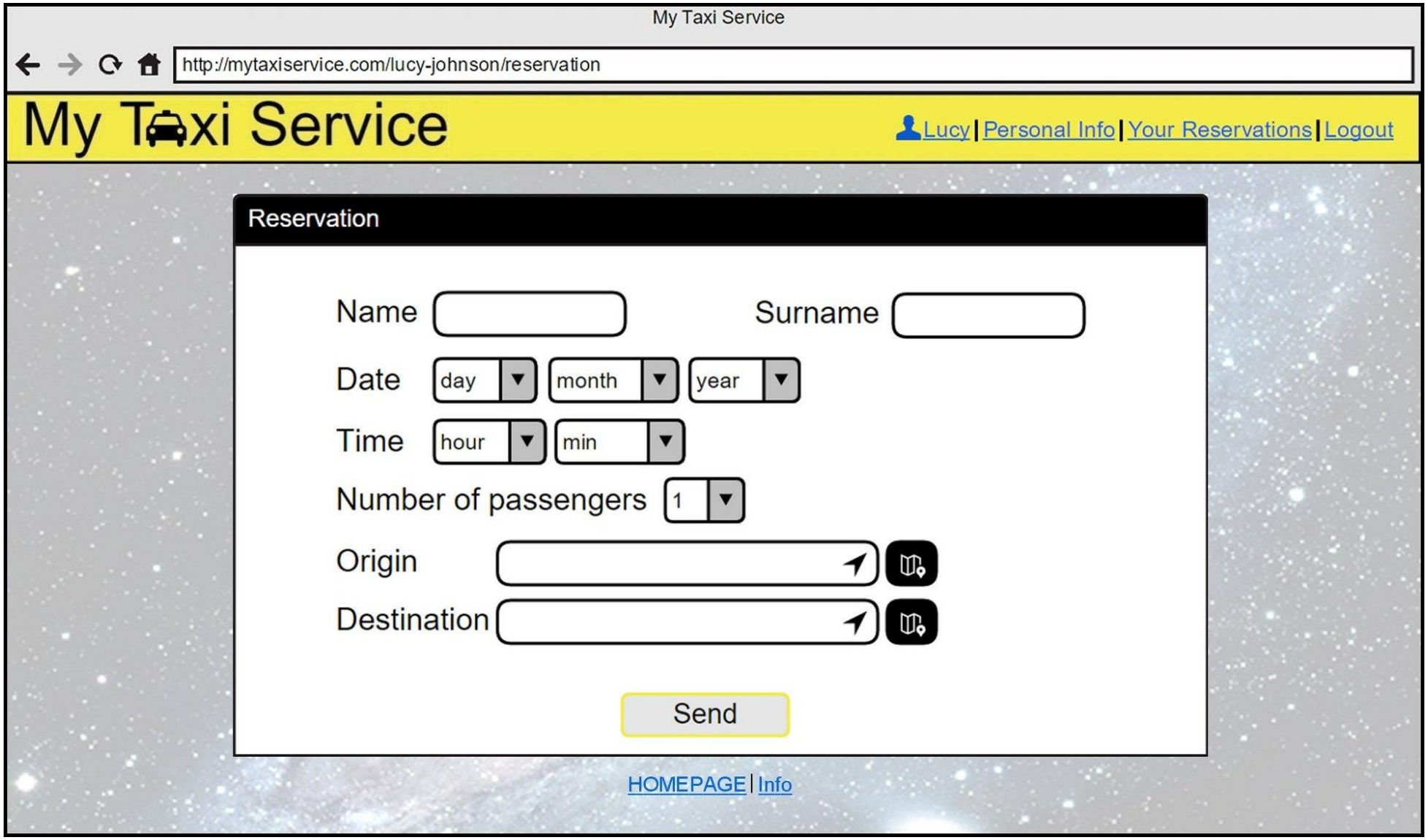
change your profile picture

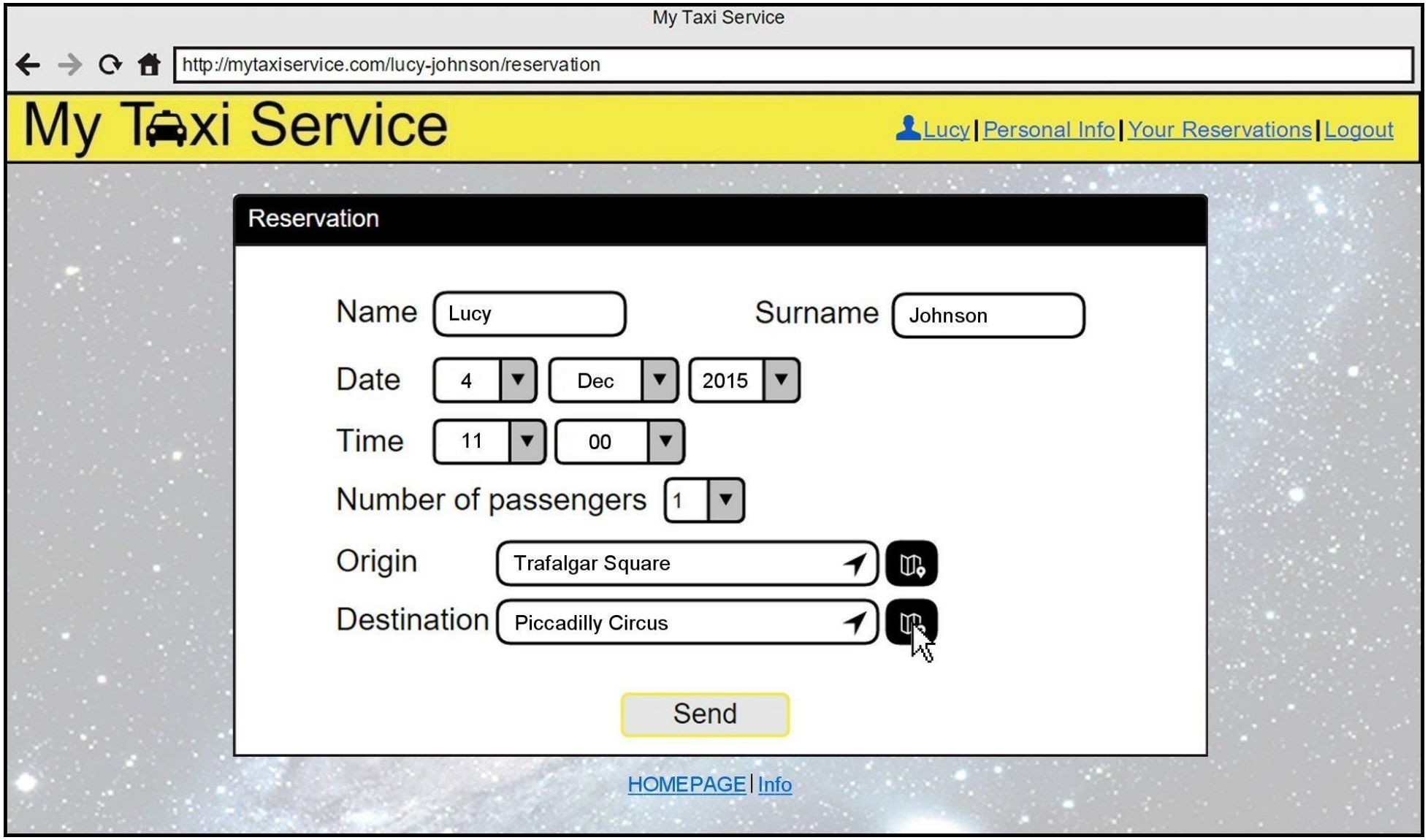
change your background image

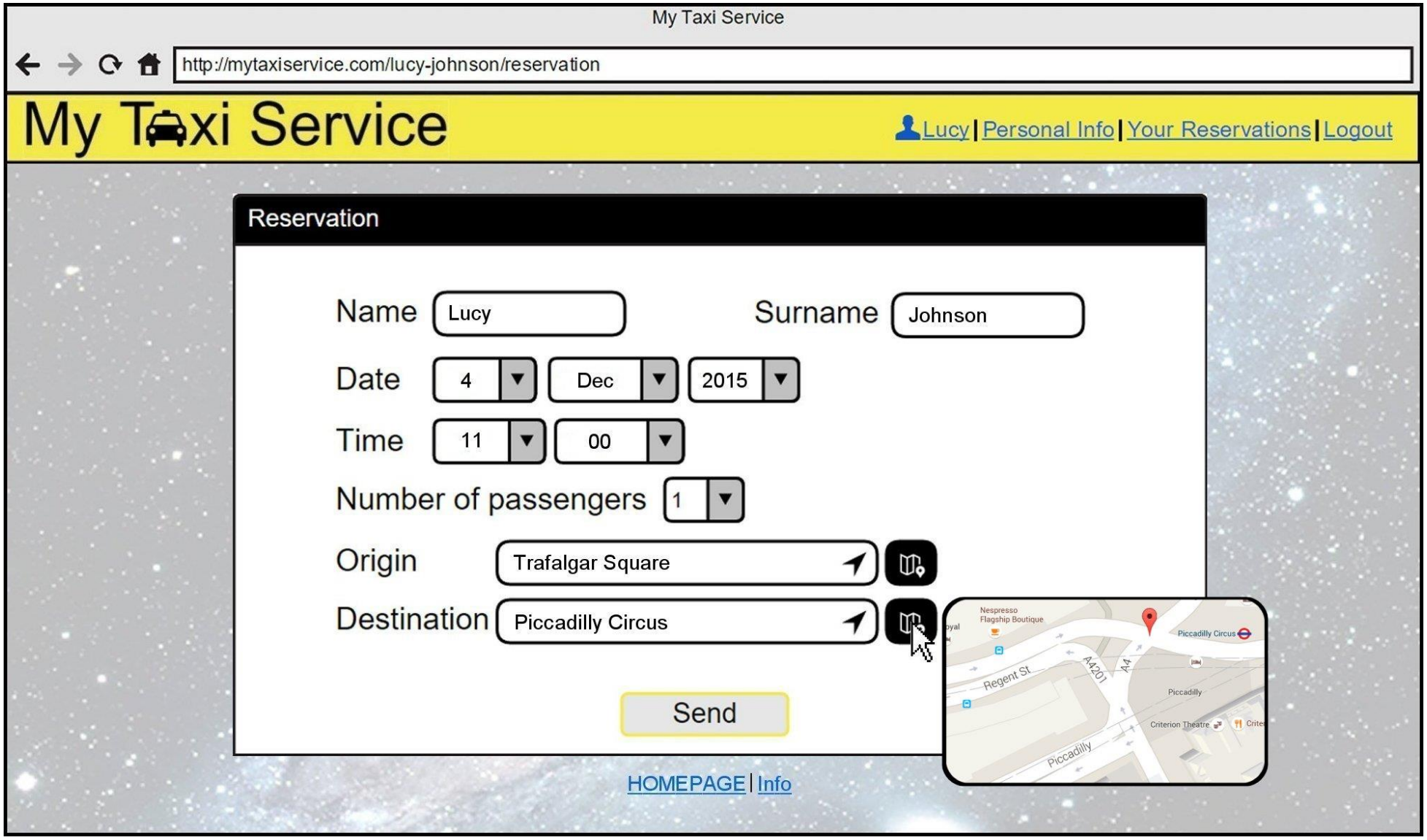
Notifications

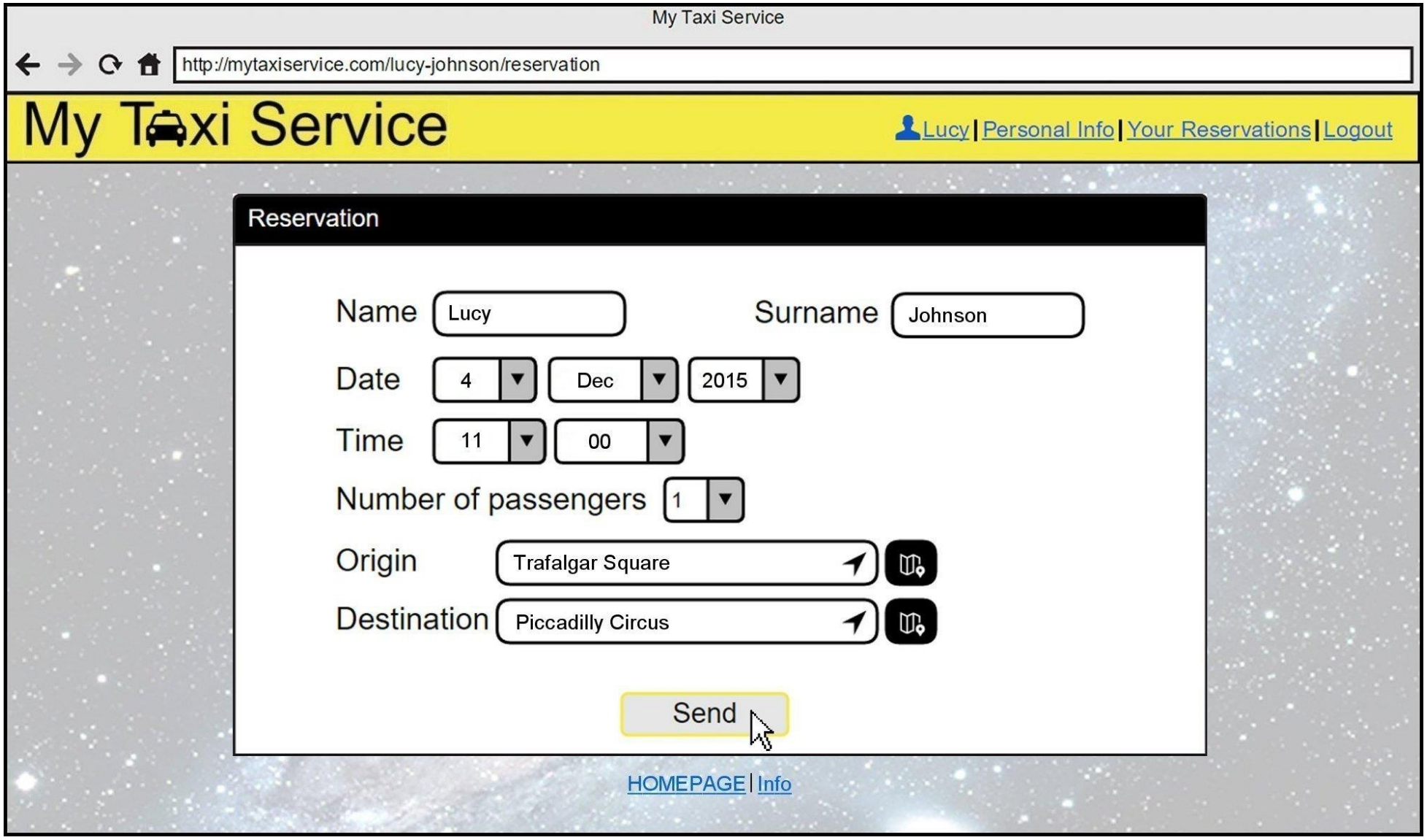
- Your request is successfully completed!
  Taxi #1125 is coming...
  Waiting Time: 6 minutes
  NEW
- Your reservation for December, 10th has been successfully registered

HOMEPAGE | Info


My Taxi Service
http://mytaxiservice.com/lucy-johnson/reservation
My Taxi Service
Lucy | Personal Info | Your Reservations | Logout
Reservation
Name
Surname
Date
day
month
year
Time
hour
min
Number of passengers
1
Origin
Destination
Send
HOMEPAGE | Info


My Taxi Service
http://mytaxiservice.com/lucy-johnson/reservation
My Taxi Service
Lucy | Personal Info | Your Reservations | Logout
Reservation
Name
Lucy
Surname
Johnson
Date
4
Dec
2015
Time
11
00
Number of passengers
1
Origin
Trafalgar Square
Destination
Piccadilly Circus
Send
HOMEPAGE | Info


My Taxi Service
http://mytaxiservice.com/lucy-johnson/reservation
My Taxi Service
Lucy | Personal Info | Your Reservations | Logout
Reservation
Name
Lucy
Surname
Johnson
Date
4
Dec
2015
Time
11
00
Number of passengers
1
Origin
Trafalgar Square
Destination
Piccadilly Circus
Send
HOMEPAGE | Info
Nespresso Flagship Boutique
Piccadilly Circus
Regent St
A4201
A4
Piccadilly
Criterion Theatre
Piccadilly


My Taxi Service
http://mytaxiservice.com/lucy-johnson/reservation
My Taxi Service
Lucy | Personal Info | Your Reservations | Logout
Reservation
Name
Lucy
Surname
Johnson
Date
4
Dec
2015
Time
11
00
Number of passengers
1
Origin
Trafalgar Square
Destination
Piccadilly Circus
Send
HOMEPAGE | Info

My Taxi Service

http://mytaxiservice.com/lucy-johnson

My Taxi Service

Lucy | Personal Info | Your Reservations | Logout

Lucy Johnson

Hi Lucy,
what do you want to do?

REQUEST

RESERVE

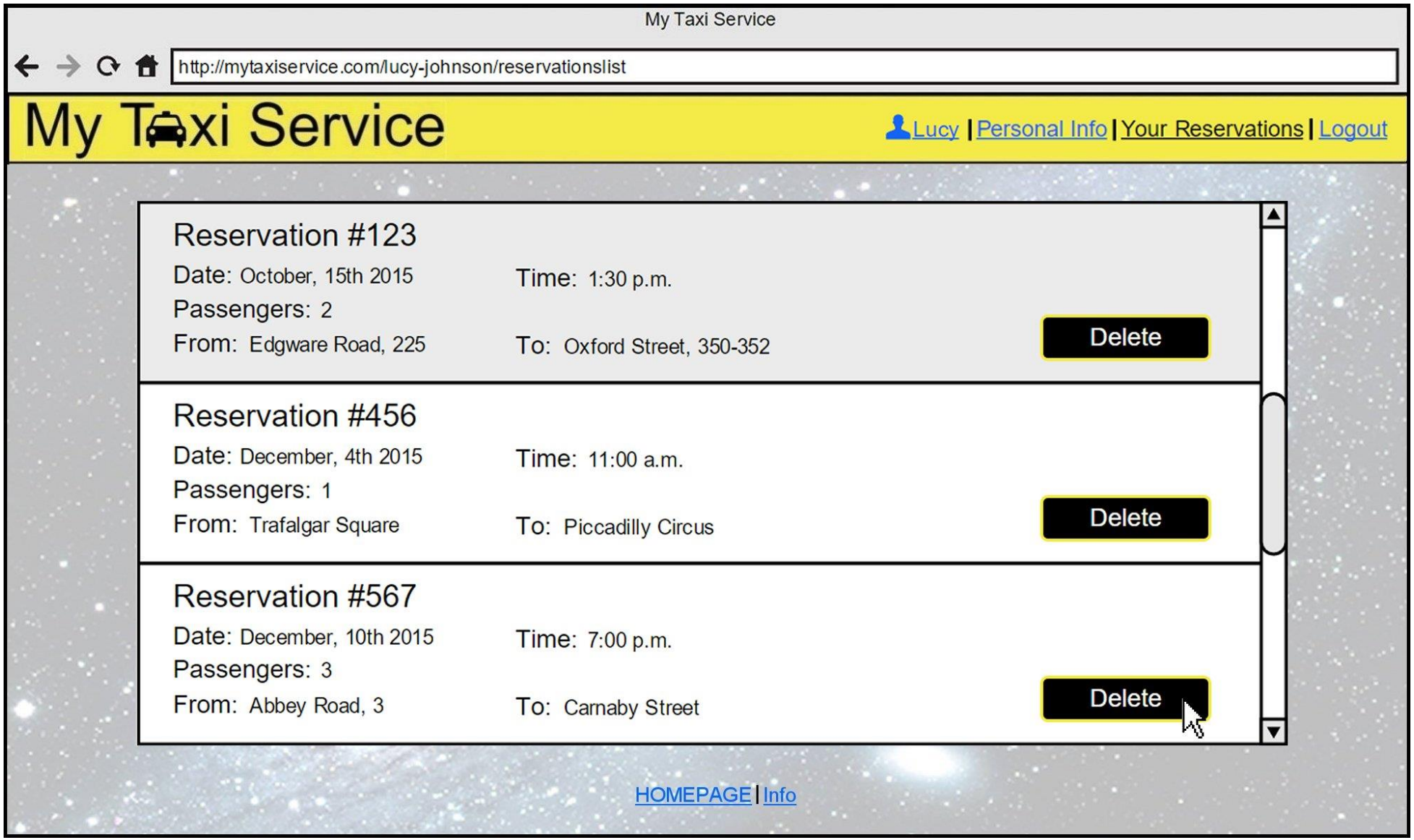
change your profile picture

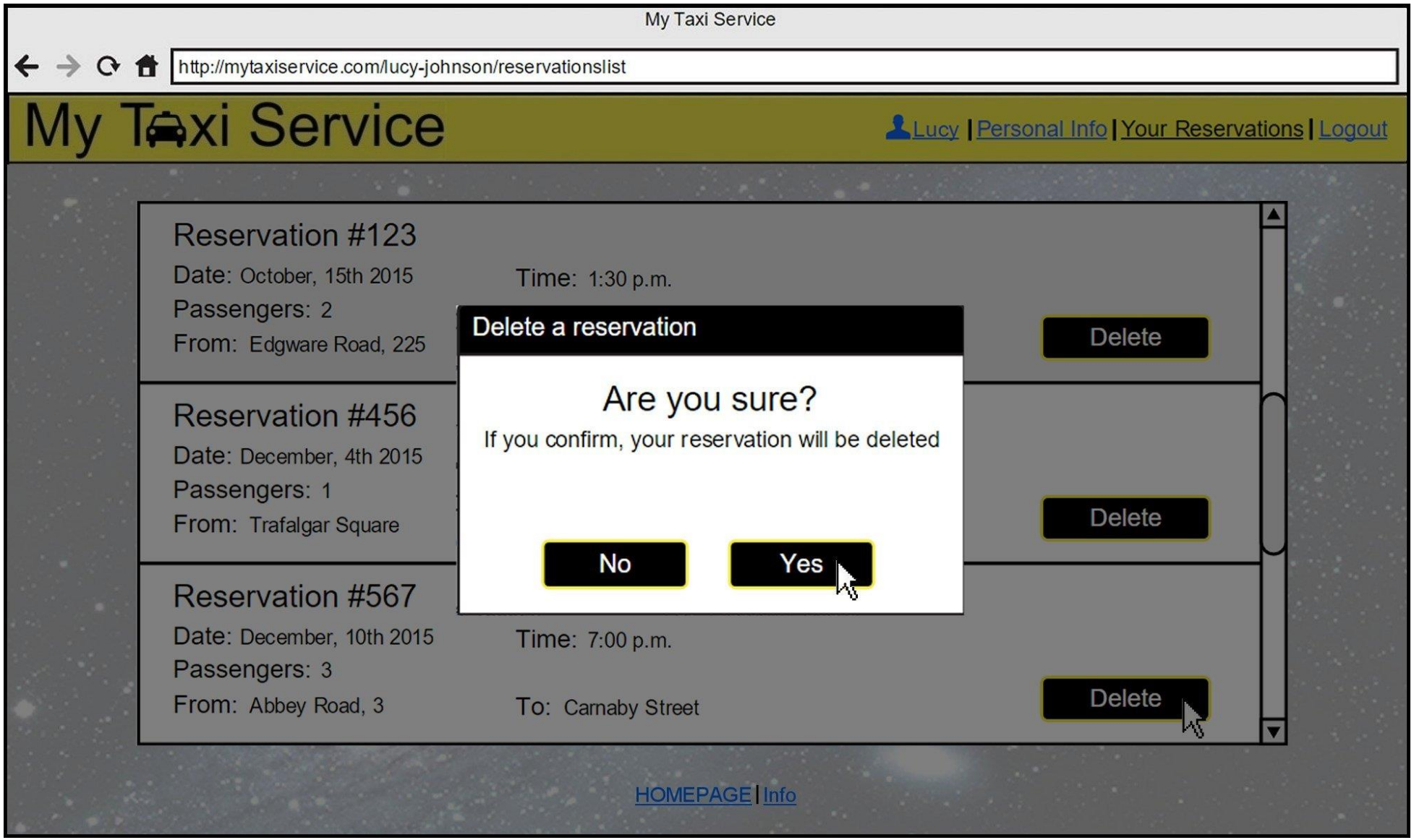
change your background image

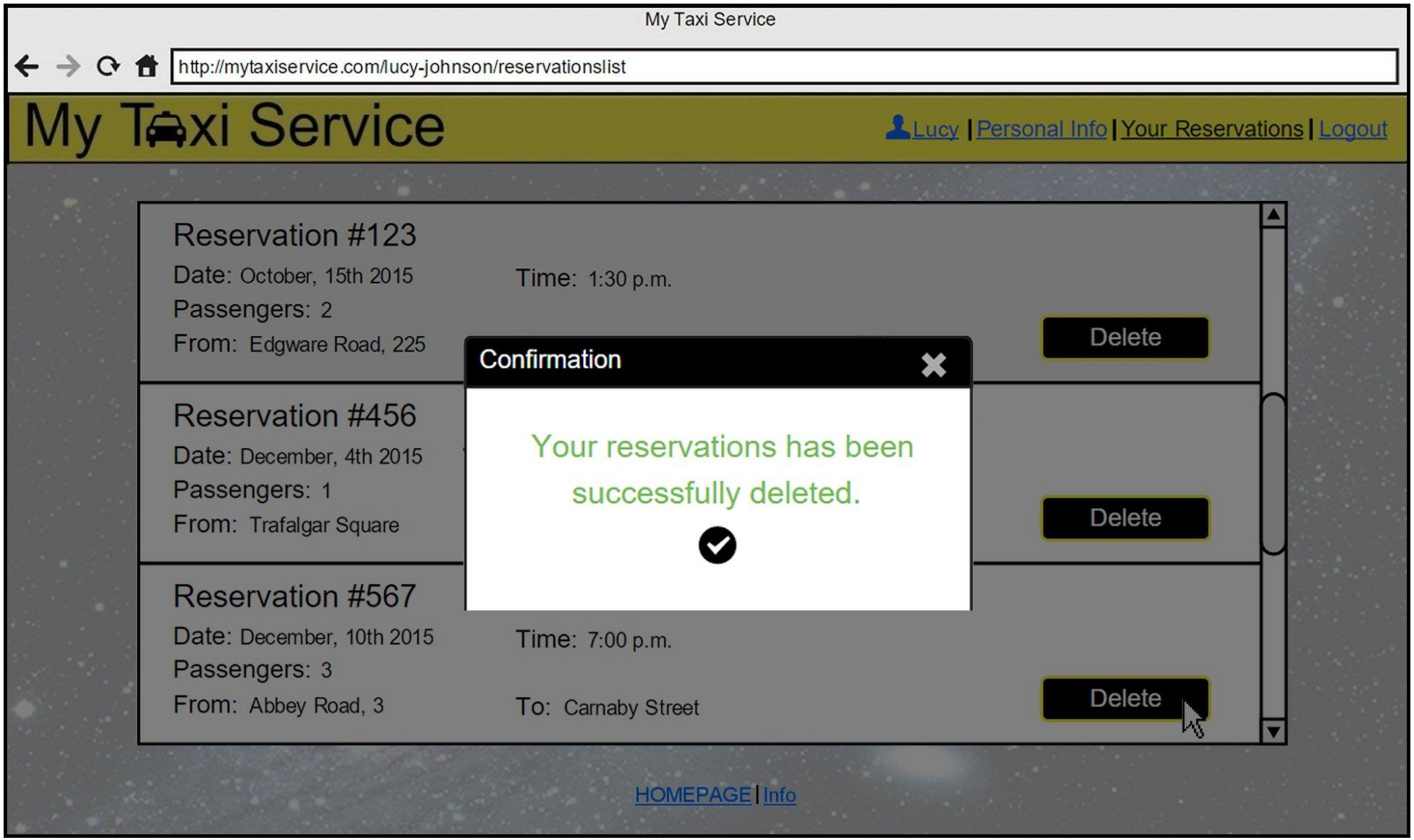
Notifications

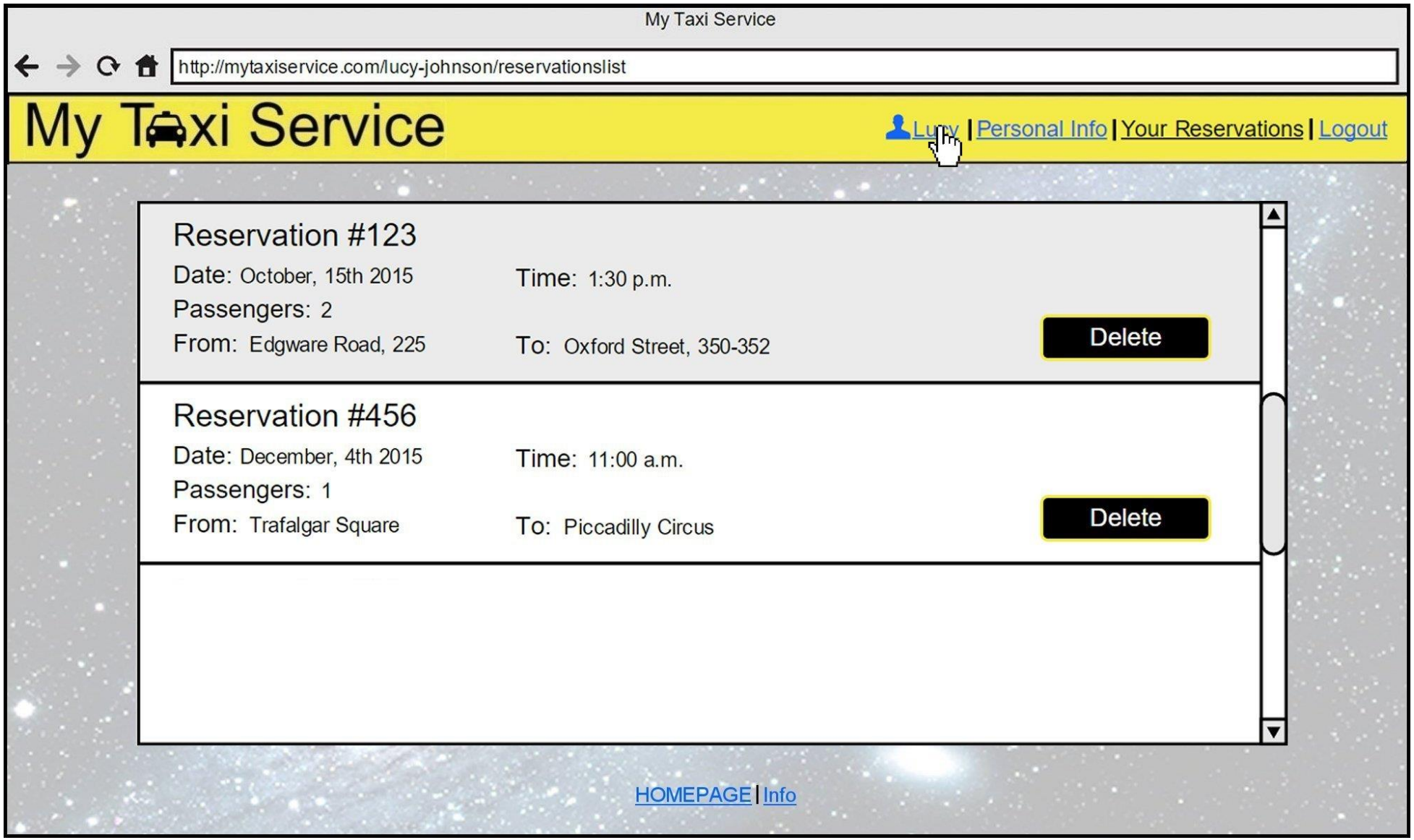
- Your reservation for December 4th has been successfully registered NEW
- Your request is successfully completed! Taxi #1125 is coming... Waiting Time: 6 minutes NEW
- Your reservation for December, 10th has been successfully registered

HOMEPAGE | Info


My Taxi Service
http://mytaxiservice.com/lucy-johnson/reservationslist
My Taxi Service
Lucy | Personal Info | Your Reservations | Logout
Reservation #123
Date: October, 15th 2015
Time: 1:30 p.m.
Passengers: 2
From: Edgware Road, 225
To: Oxford Street, 350-352
Delete
Reservation #456
Date: December, 4th 2015
Time: 11:00 a.m.
Passengers: 1
From: Trafalgar Square
To: Piccadilly Circus
Delete
Reservation #567
Date: December, 10th 2015
Time: 7:00 p.m.
Passengers: 3
From: Abbey Road, 3
To: Carnaby Street
Delete
HOMEPAGE | Info


My Taxi Service
http://mytaxiservice.com/lucy-johnson/reservationslist
My Taxi Service
Lucy | Personal Info | Your Reservations | Logout
Reservation #123
Date: October, 15th 2015
Passengers: 2
From: Edgware Road, 225
Time: 1:30 p.m.
Delete
Reservation #456
Date: December, 4th 2015
Passengers: 1
From: Trafalgar Square
Delete
Reservation #567
Date: December, 10th 2015
Passengers: 3
From: Abbey Road, 3
Time: 7:00 p.m.
To: Carnaby Street
Delete
Delete a reservation
Are you sure?
If you confirm, your reservation will be deleted
No
Yes
HOMEPAGE | Info


My Taxi Service
http://mytaxiservice.com/lucy-johnson/reservationslist
My Taxi Service
Lucy | Personal Info | Your Reservations | Logout
Reservation #123
Date: October, 15th 2015
Time: 1:30 p.m.
Passengers: 2
From: Edgware Road, 225
Delete
Confirmation
Your reservations has been successfully deleted.
Reservation #456
Date: December, 4th 2015
Passengers: 1
From: Trafalgar Square
Delete
Reservation #567
Date: December, 10th 2015
Time: 7:00 p.m.
Passengers: 3
From: Abbey Road, 3
To: Carnaby Street
Delete
HOMEPAGE | Info


My Taxi Service
http://mytaxiservice.com/lucy-johnson/reservationslist
My Taxi Service
Lucy | Personal Info | Your Reservations | Logout
Reservation #123
Date: October, 15th 2015
Time: 1:30 p.m.
Passengers: 2
From: Edgware Road, 225
To: Oxford Street, 350-352
Delete
Reservation #456
Date: December, 4th 2015
Time: 11:00 a.m.
Passengers: 1
From: Trafalgar Square
To: Piccadilly Circus
Delete
HOMEPAGE | Info

My Taxi Service

http://mytaxiservice.com/lucy-johnson

My Taxi Service

Lucy | Personal Info | Your Reservations | Logout

Lucy Johnson

Hi Lucy,
what do you want to do?

REQUEST

RESERVE

change your profile picture

change your background image

Notifications

- Your reservation for December 10th has been deleted NEW
- Your reservation for December 4th has been successfully registered NEW
- Your request is successfully completed! Taxi #1125 is coming... Waiting Time: 6 minutes NEW
- Your reservation for December, 10th has been successfully registered

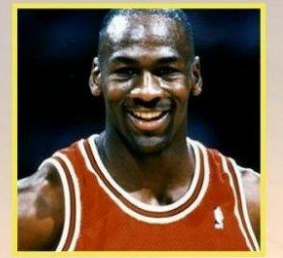
HOMEPAGE | Info

My Taxi Service

http://mytaxiservice.com/john-jones

My Taxi Service

John | Personal Info | Your Reservations | Logout



John Jones

Hi John,
what do you want to do?

REQUEST

RESERVE

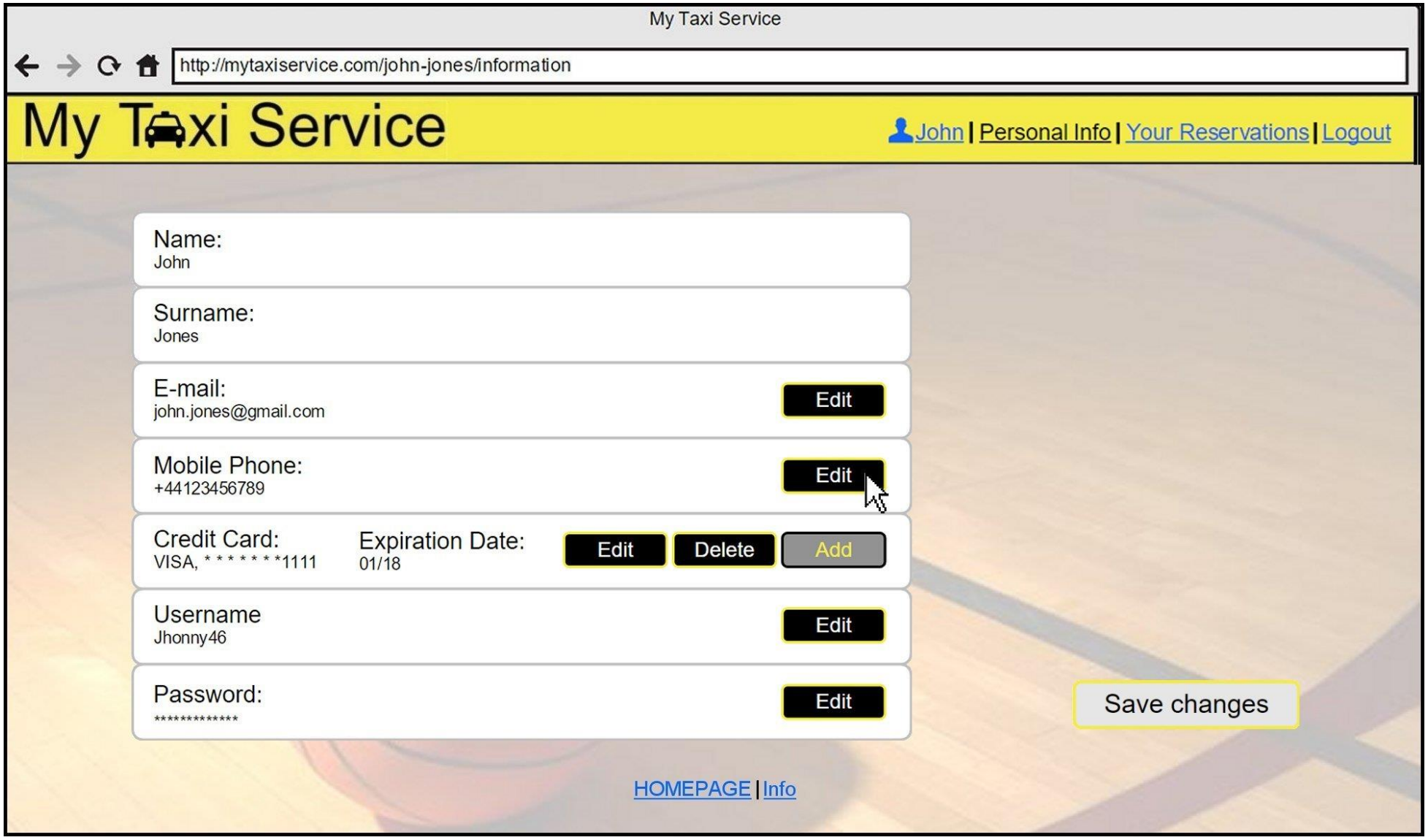
change your profile picture

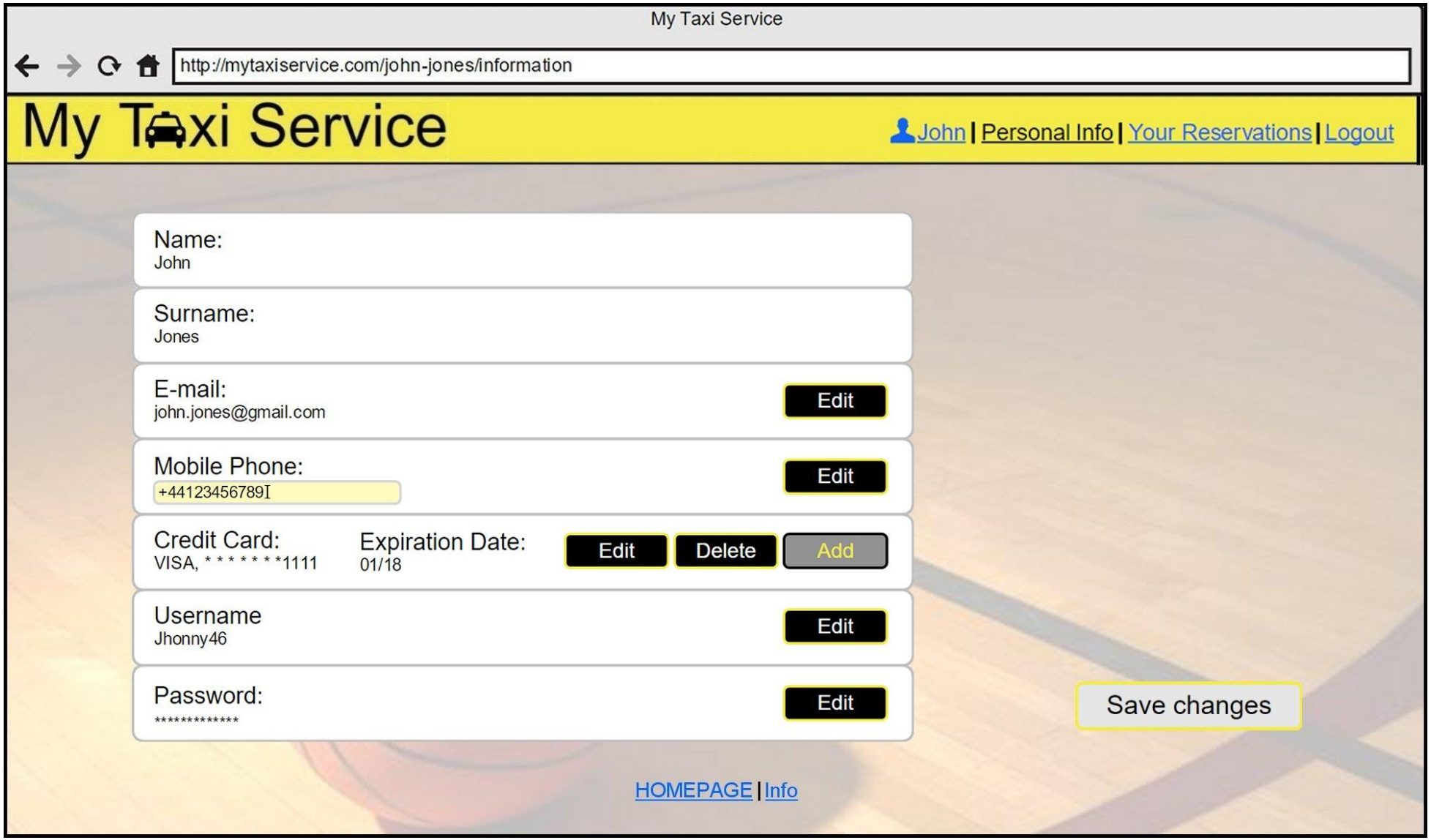
change your backgroud image

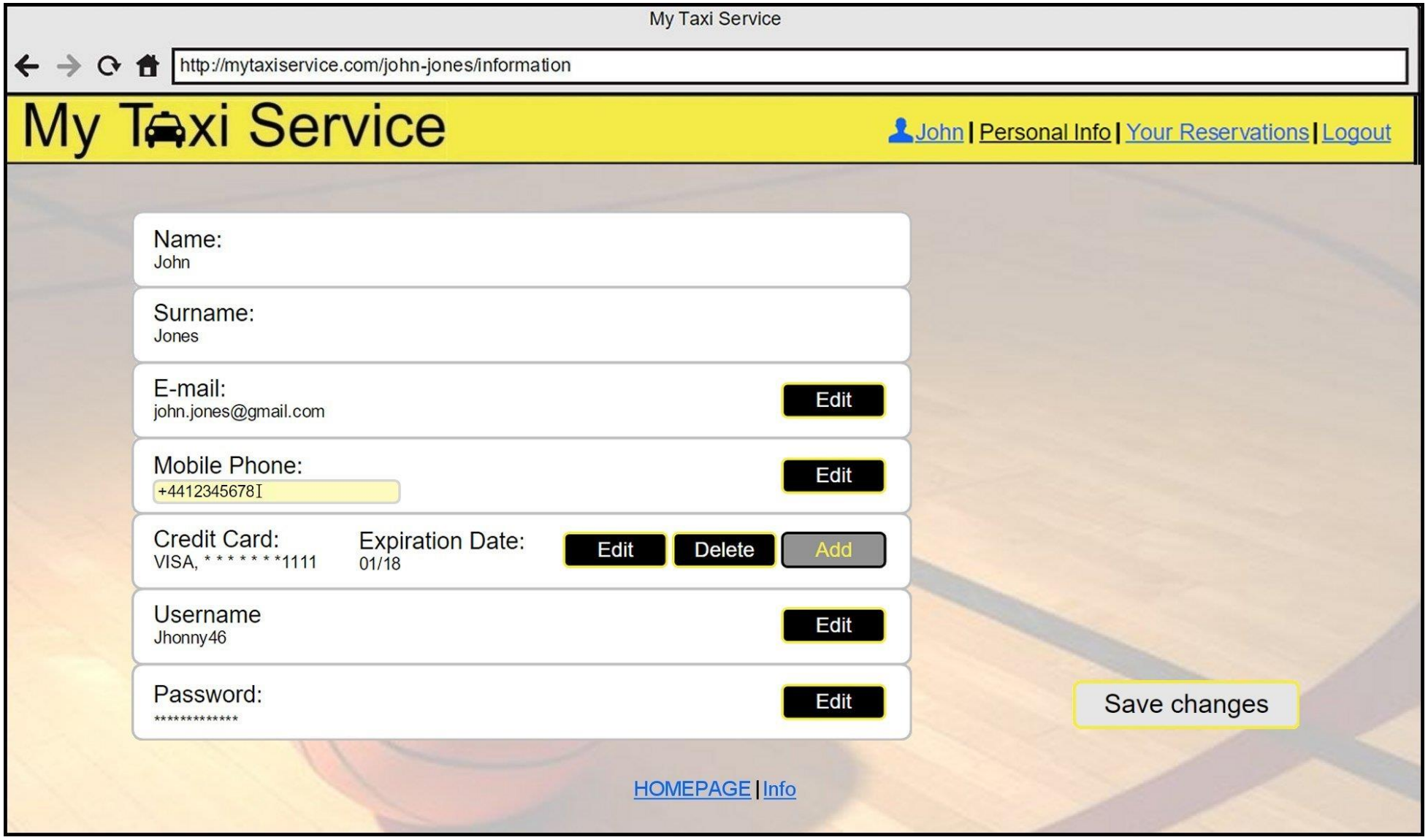
Notifications

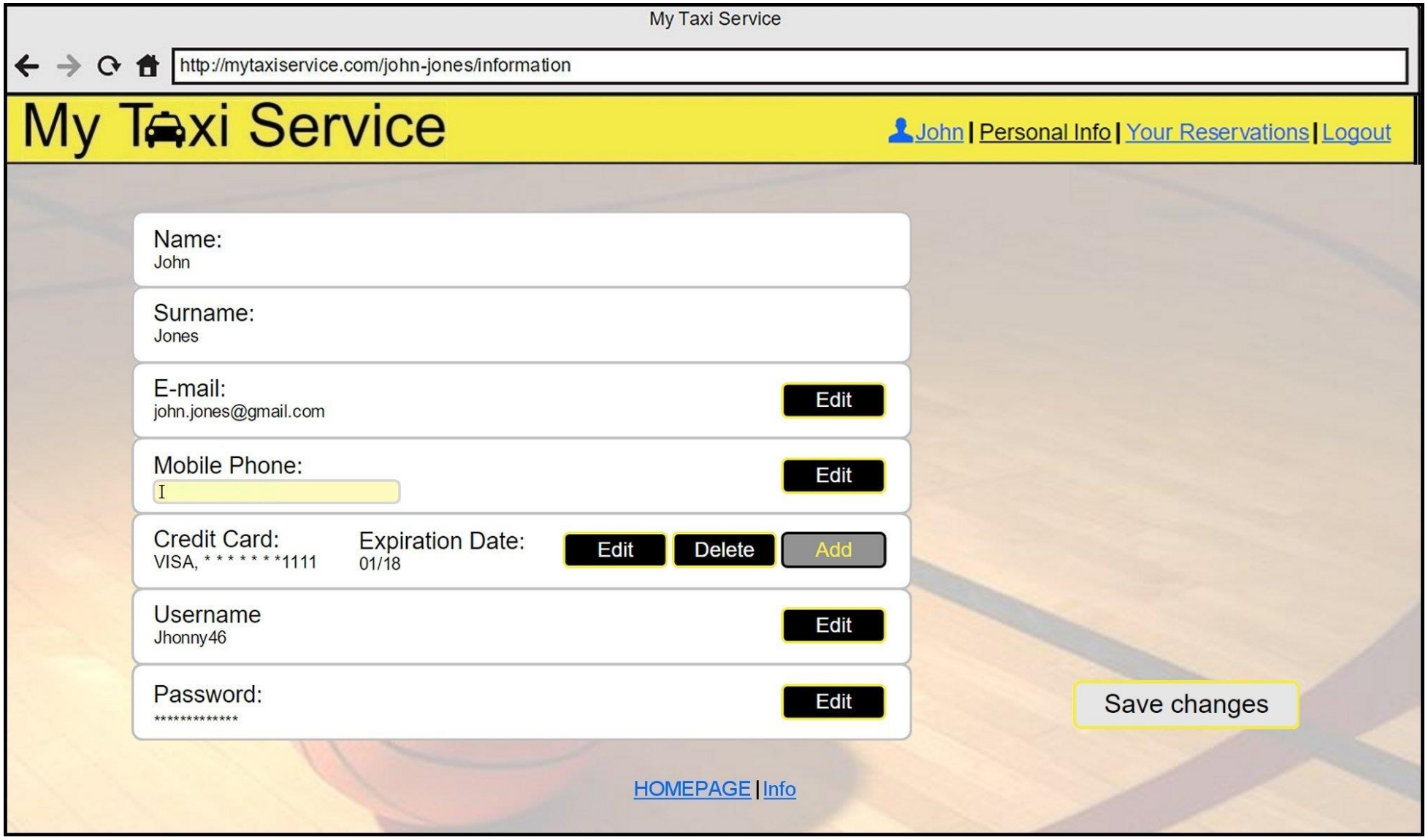
- Your request is successfully completed!
  Taxi #1023 is coming...
  Waiting Time: 2 minutes
  NEW
- Your reservation for tomorrow
  has been successfully registered

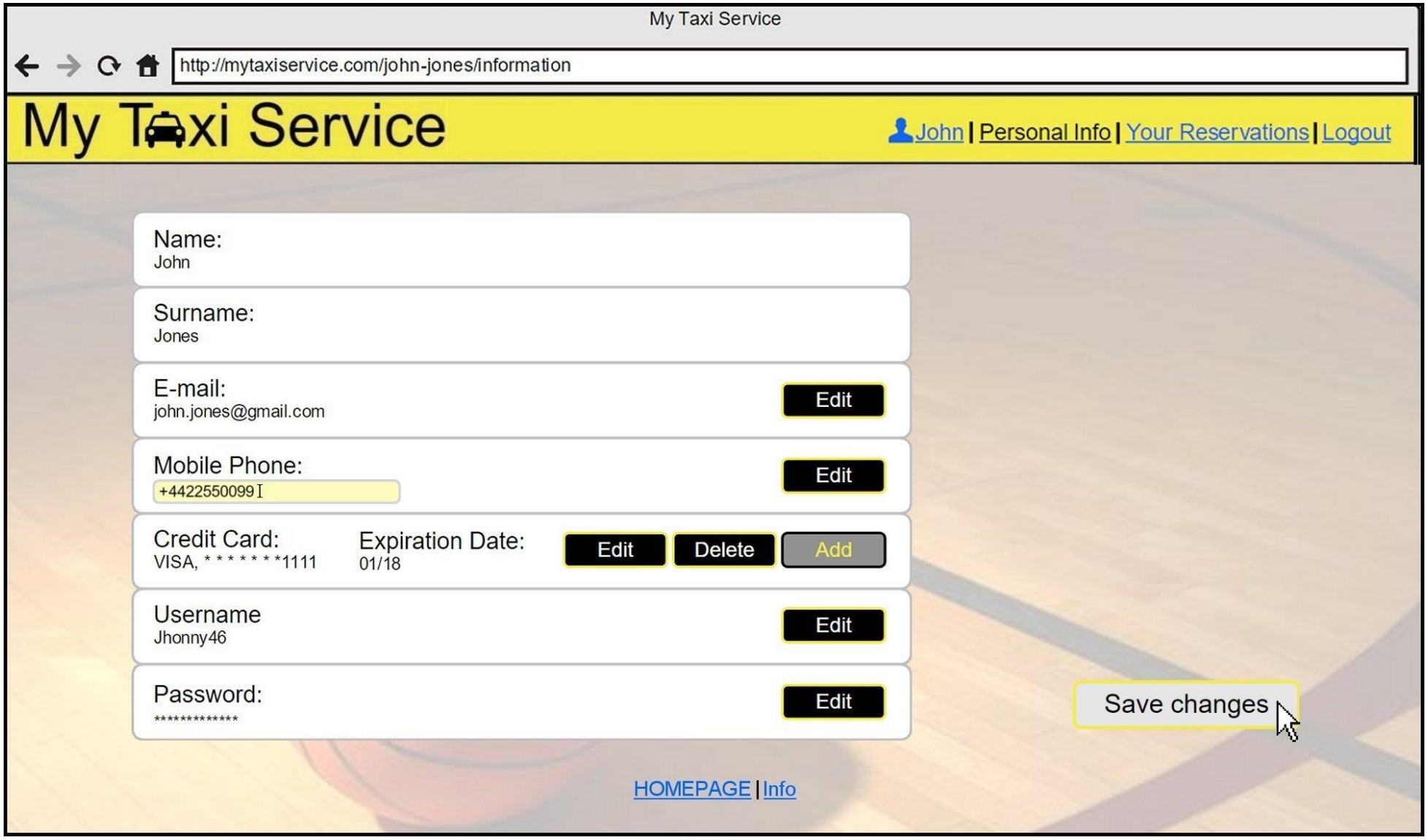
HOMEPAGE | Info

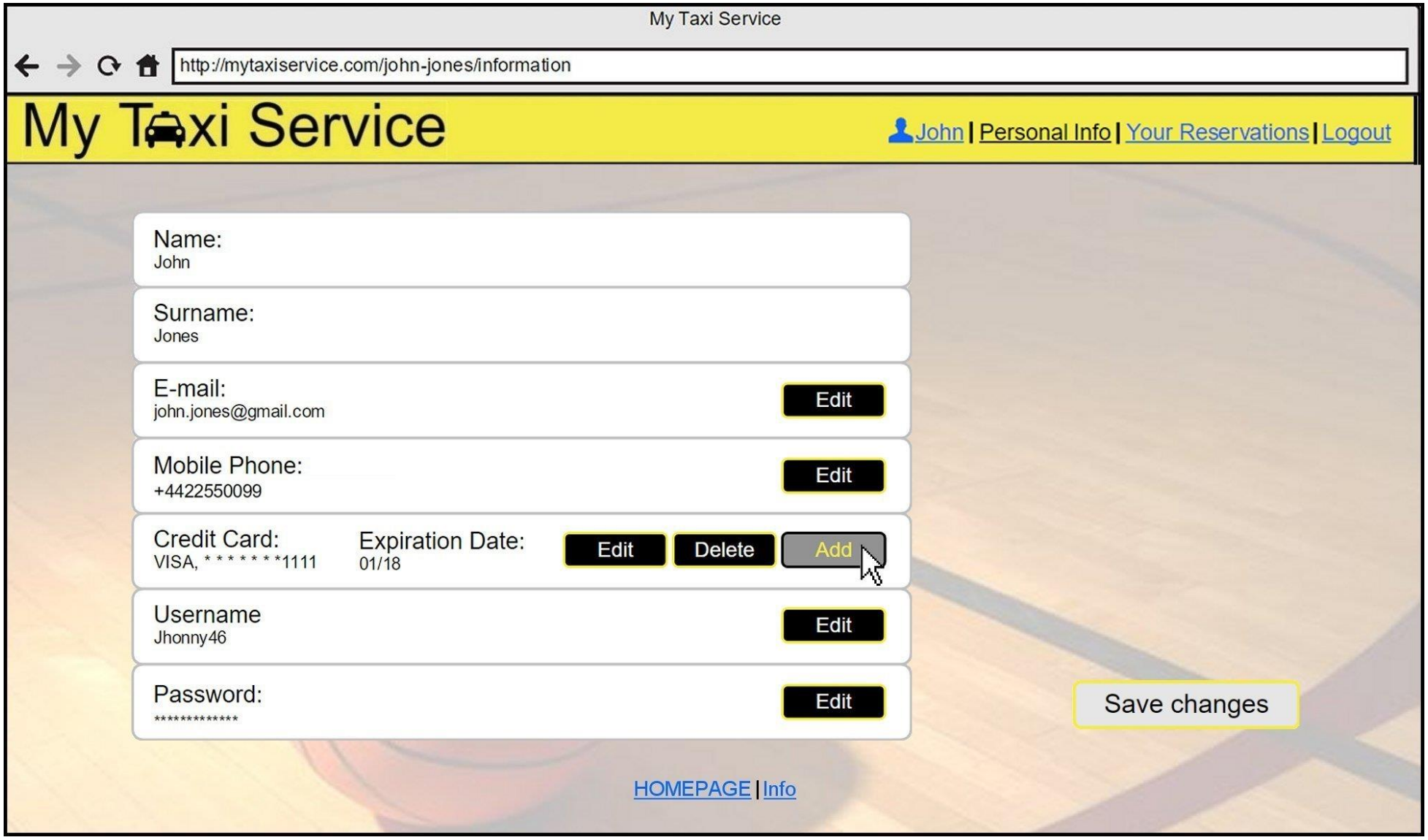

My Taxi Service
http://mytaxiservice.com/john-jones/information
My Taxi Service
John | Personal Info | Your Reservations | Logout
Name:
John
Surname:
Jones
E-mail:
john.jones@gmail.com
Edit
Mobile Phone:
+44123456789
Edit
Credit Card:
VISA, * * * * * * *1111
Expiration Date:
01/18
Edit
Delete
Add
Username
Jhonny46
Edit
Password:
****
Edit
Save changes
HOMEPAGE | Info

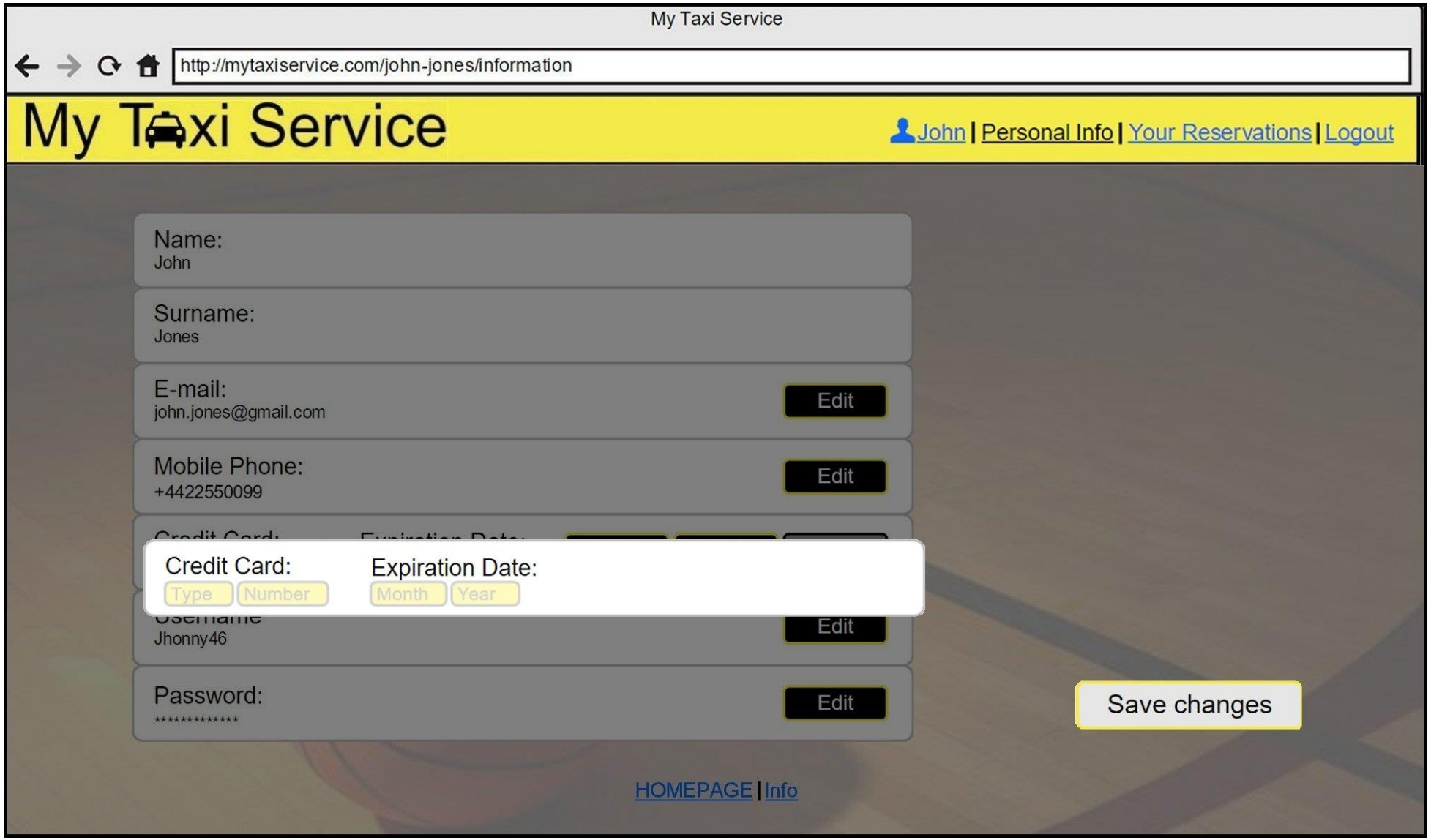

My Taxi Service
http://mytaxiservice.com/john-jones/information
My Taxi Service
John | Personal Info | Your Reservations | Logout
Name:
John
Surname:
Jones
E-mail:
john.jones@gmail.com
Edit
Mobile Phone:
+44123456789
Edit
Credit Card:
VISA, * * * * * * *1111
Expiration Date:
01/18
Edit
Delete
Add
Username
Jhonny46
Edit
Password:
****
Edit
Save changes
HOMEPAGE | Info

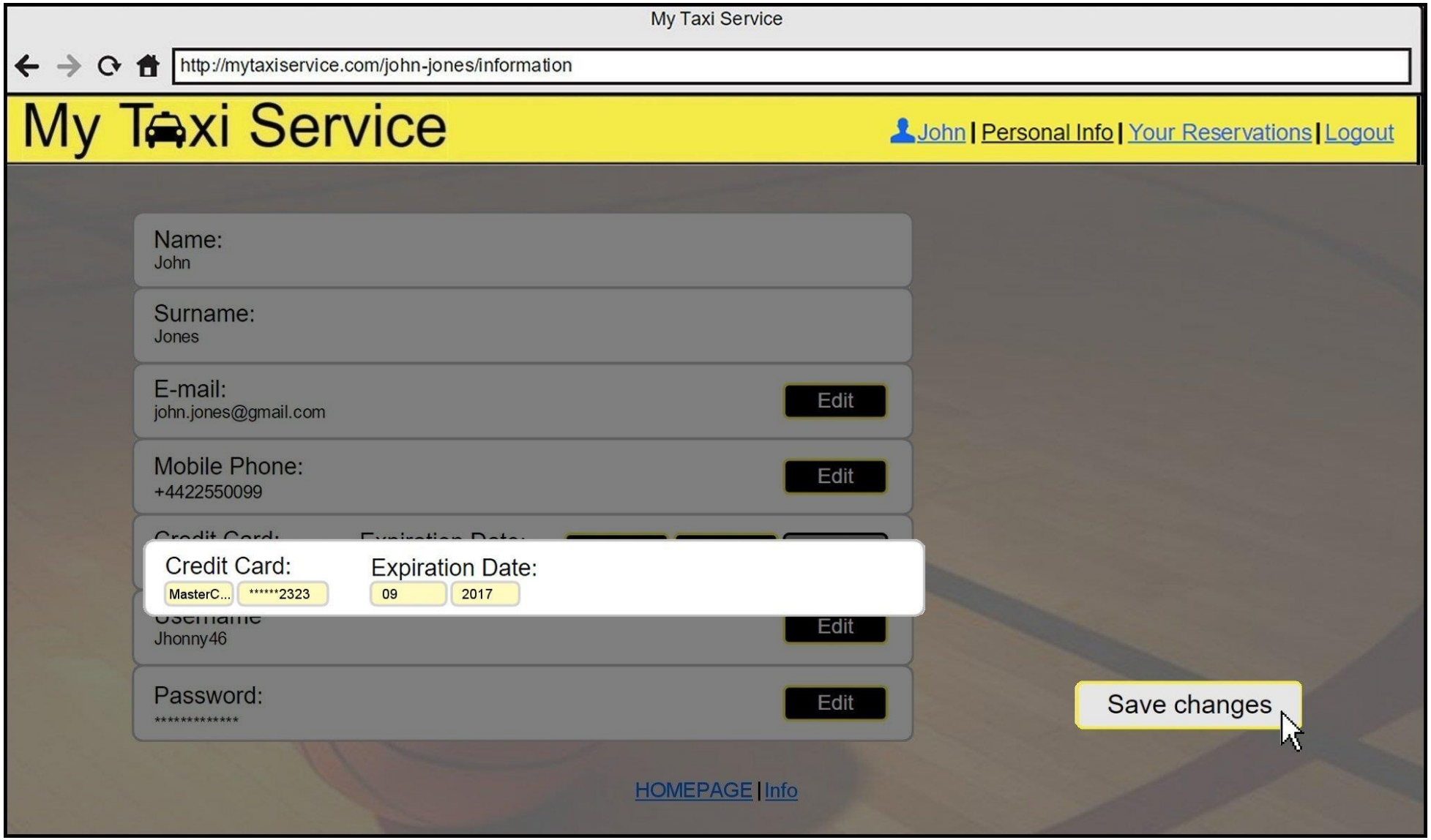

My Taxi Service
http://mytaxiservice.com/john-jones/information
My Taxi Service
John | Personal Info | Your Reservations | Logout
Name:
John
Surname:
Jones
E-mail:
john.jones@gmail.com
Edit
Mobile Phone:
+4412345678
Edit
Credit Card:
VISA, * * * * * * *1111
Expiration Date:
01/18
Edit
Delete
Add
Username
Jhonny46
Edit
Password:
****
Edit
Save changes
HOMEPAGE | Info

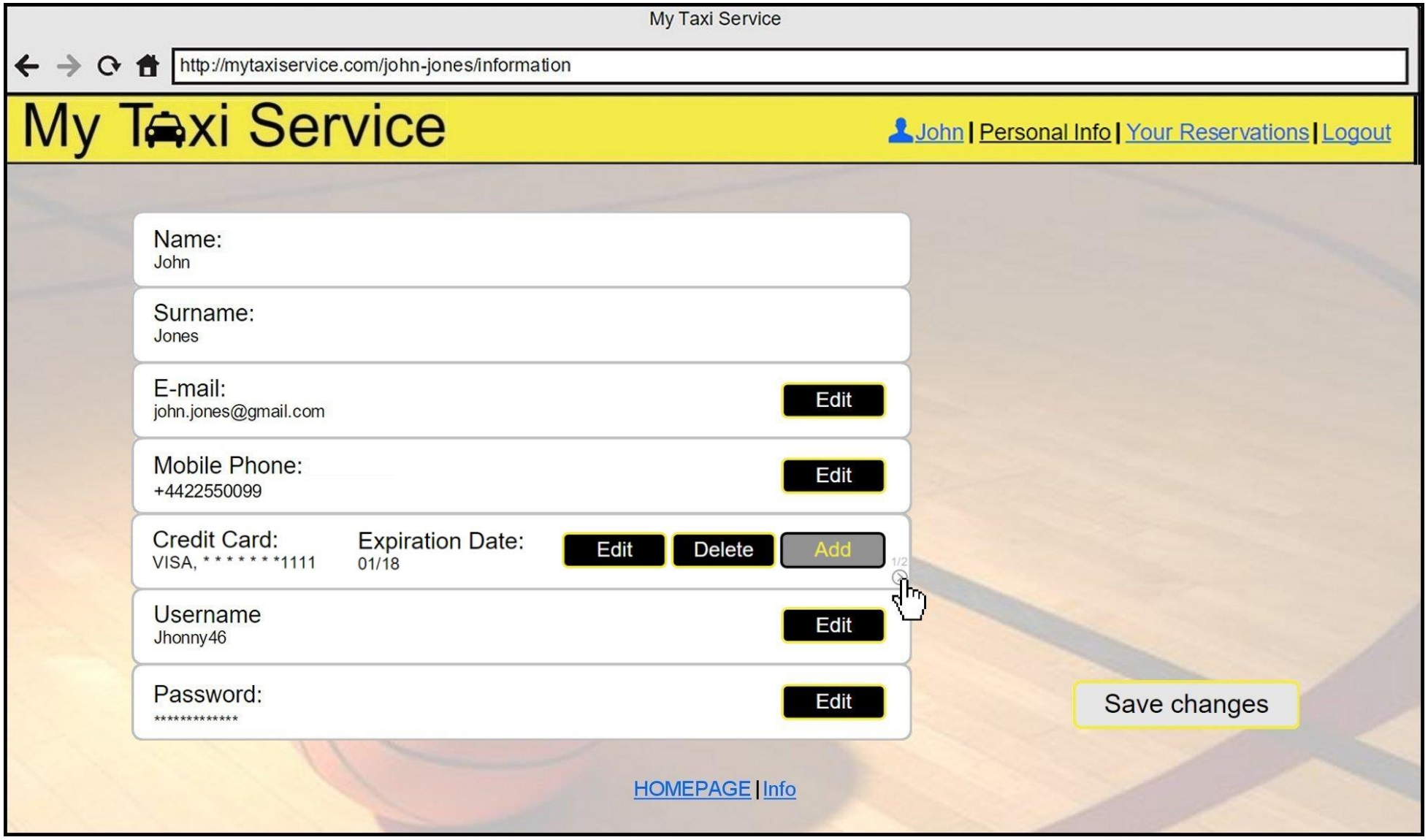

My Taxi Service
http://mytaxiservice.com/john-jones/information
My Taxi Service
John | Personal Info | Your Reservations | Logout
Name:
John
Surname:
Jones
E-mail:
john.jones@gmail.com
Edit
Mobile Phone:
Edit
Credit Card:
VISA, * * * * * * *1111
Expiration Date:
01/18
Edit
Delete
Add
Username
Jhonny46
Edit
Password:
****
Edit
Save changes
HOMEPAGE | Info

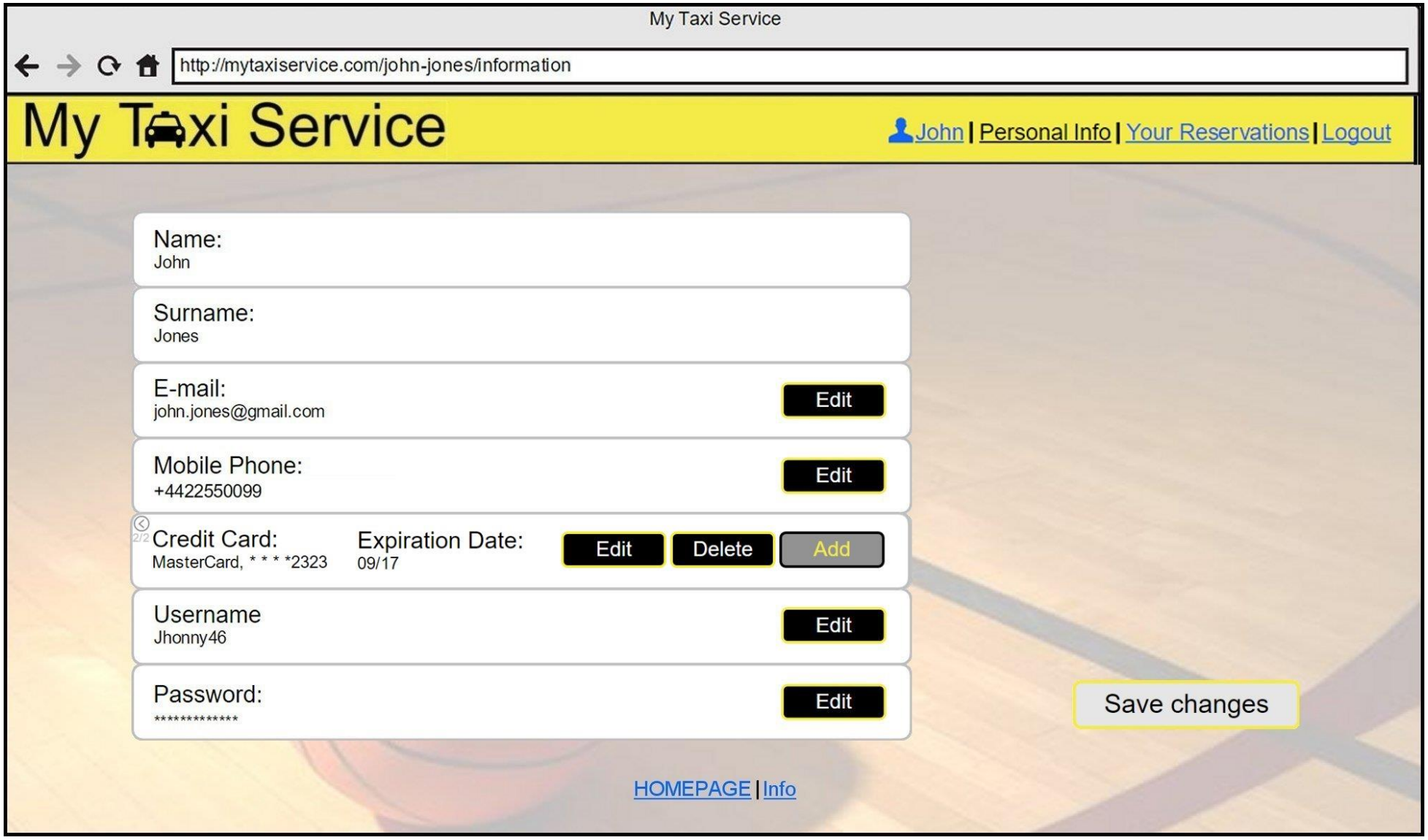

My Taxi Service
http://mytaxiservice.com/john-jones/information
My Taxi Service
John | Personal Info | Your Reservations | Logout
Name:
John
Surname:
Jones
E-mail:
john.jones@gmail.com
Edit
Mobile Phone:
+4422550099
Edit
Credit Card:
VISA, * * * * * * *1111
Expiration Date:
01/18
Edit
Delete
Add
Username
Jhonny46
Edit
Password:
****
Edit
Save changes
HOMEPAGE | Info


My Taxi Service
http://mytaxiservice.com/john-jones/information
My Taxi Service
John | Personal Info | Your Reservations | Logout
Name:
John
Surname:
Jones
E-mail:
john.jones@gmail.com
Edit
Mobile Phone:
+4422550099
Edit
Credit Card:
VISA, * * * * * * *1111
Expiration Date:
01/18
Edit
Delete
Add
Username
Jhonny46
Edit
Password:
****
Edit
Save changes
HOMEPAGE | Info

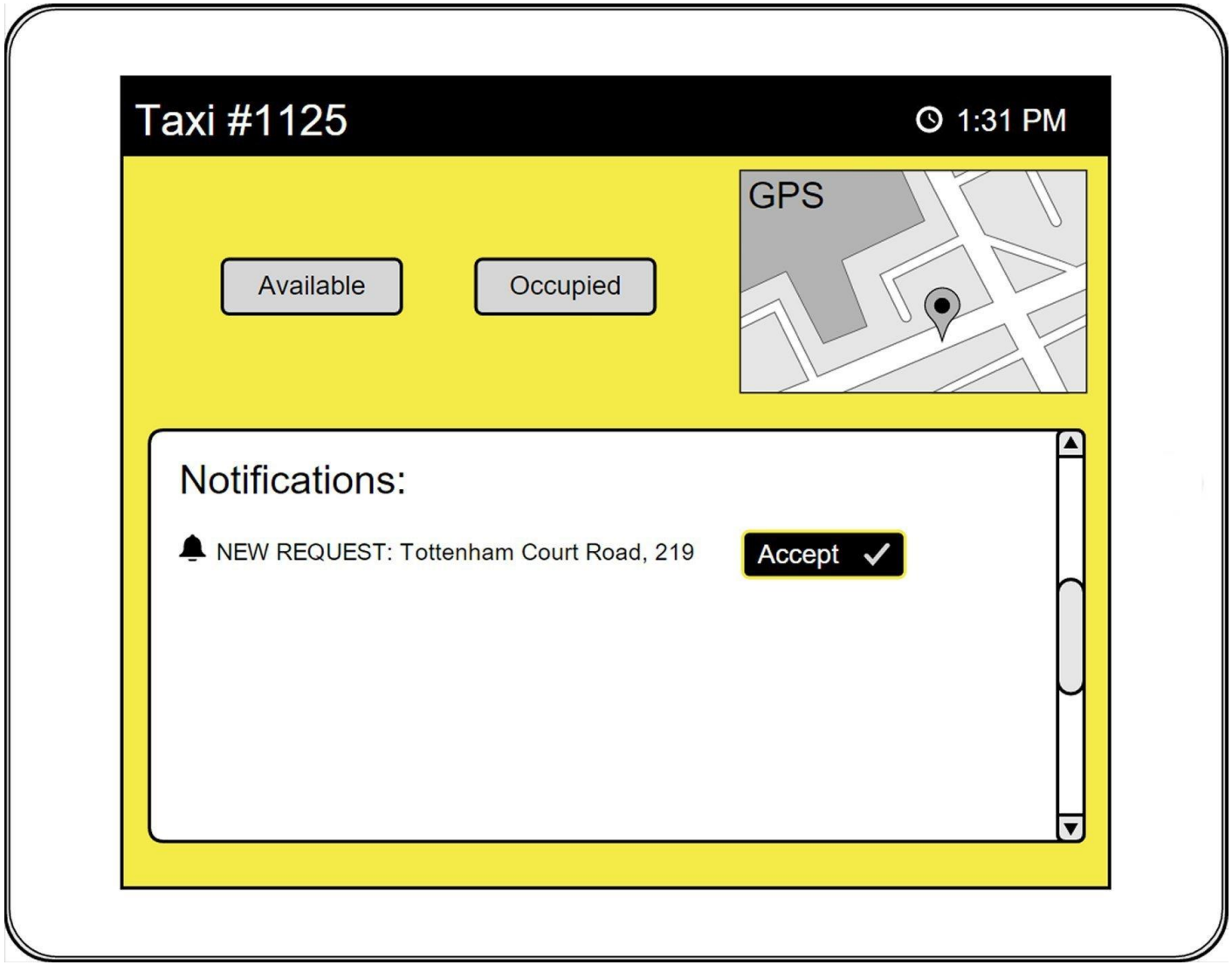

My Taxi Service
http://mytaxiservice.com/john-jones/information
My Taxi Service
John | Personal Info | Your Reservations | Logout
Name:
John
Surname:
Jones
E-mail:
john.jones@gmail.com
Edit
Mobile Phone:
+4422550099
Edit
Credit Card:
Type
Number
Expiration Date:
Month
Year
Edit
Jhonny46
Password:
****
Edit
Save changes
HOMEPAGE | Info

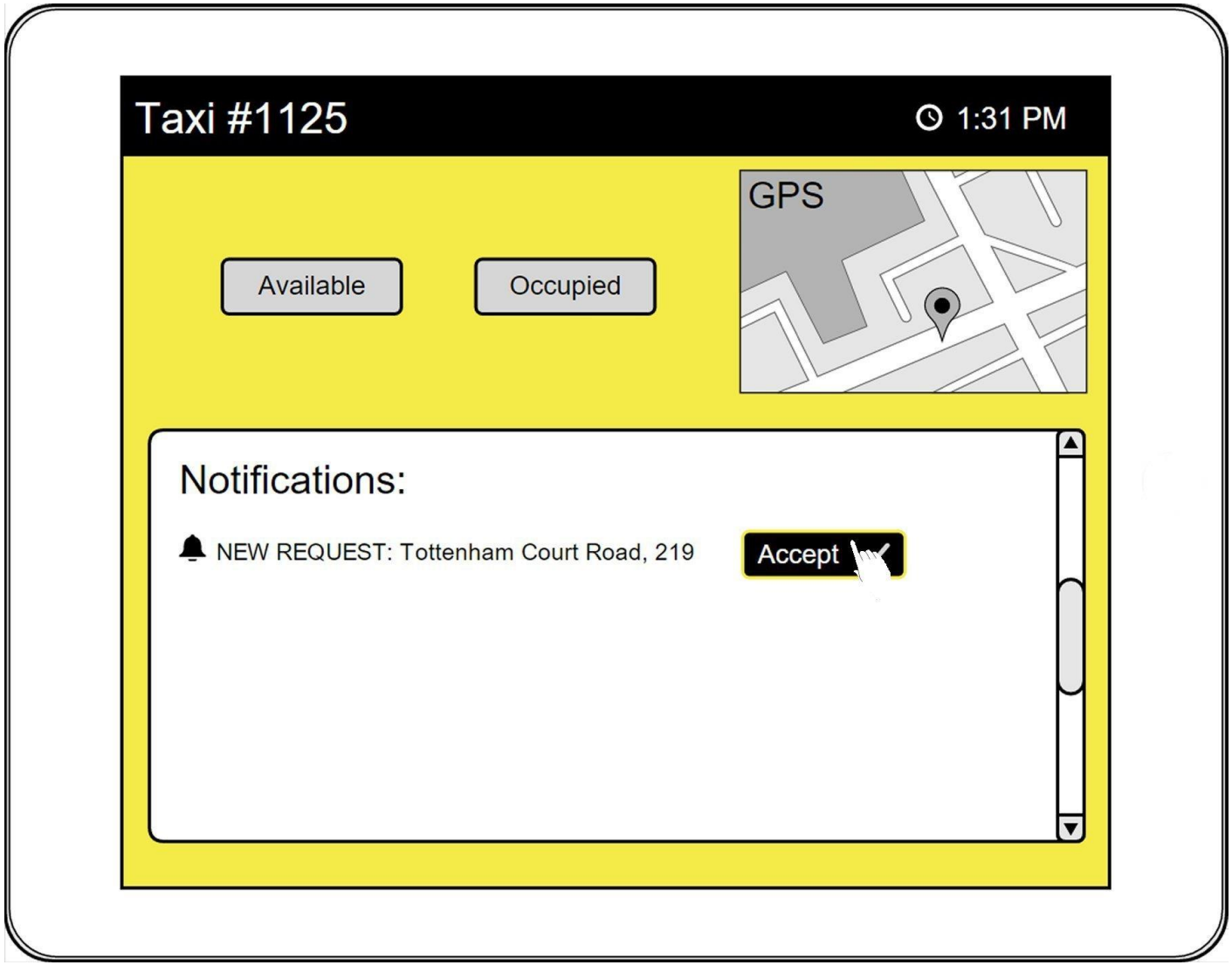

My Taxi Service
http://mytaxiservice.com/john-jones/information
My Taxi Service
John | Personal Info | Your Reservations | Logout
Name:
John
Surname:
Jones
E-mail:
john.jones@gmail.com
Edit
Mobile Phone:
+4422550099
Edit
Credit Card:
MasterC...
*****2323
Expiration Date:
09
2017
Jhonny46
Edit
Password:
****
Edit
Save changes
HOMEPAGE | Info

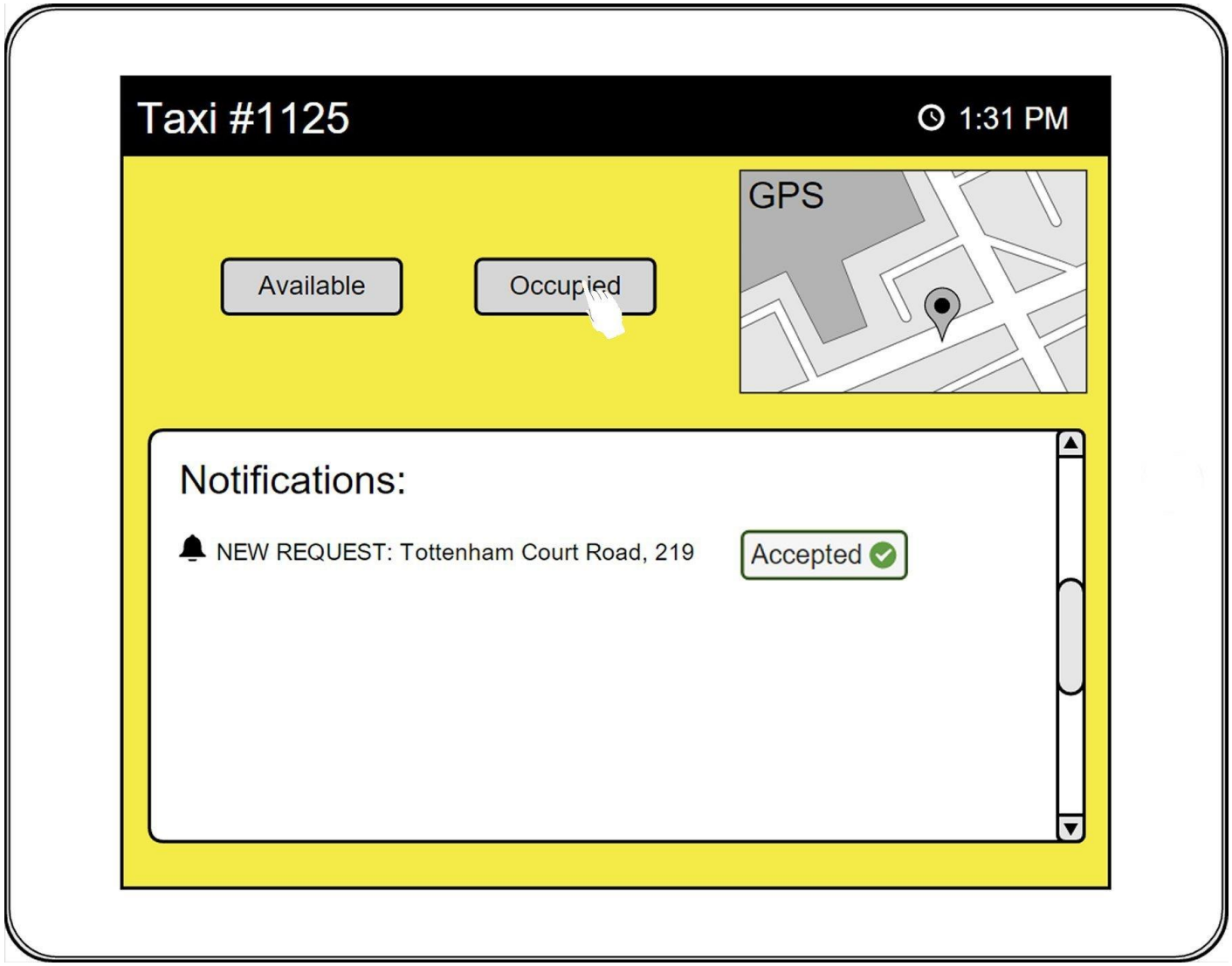

My Taxi Service
http://mytaxiservice.com/john-jones/information
My Taxi Service
John | Personal Info | Your Reservations | Logout
Name:
John
Surname:
Jones
E-mail:
john.jones@gmail.com
Edit
Mobile Phone:
+4422550099
Edit
Credit Card:
VISA, * * * * * * *1111
Expiration Date:
01/18
Edit
Delete
Add
1/2
Username
Jhonny46
Edit
Password:
****
Edit
Save changes
HOMEPAGE | Info

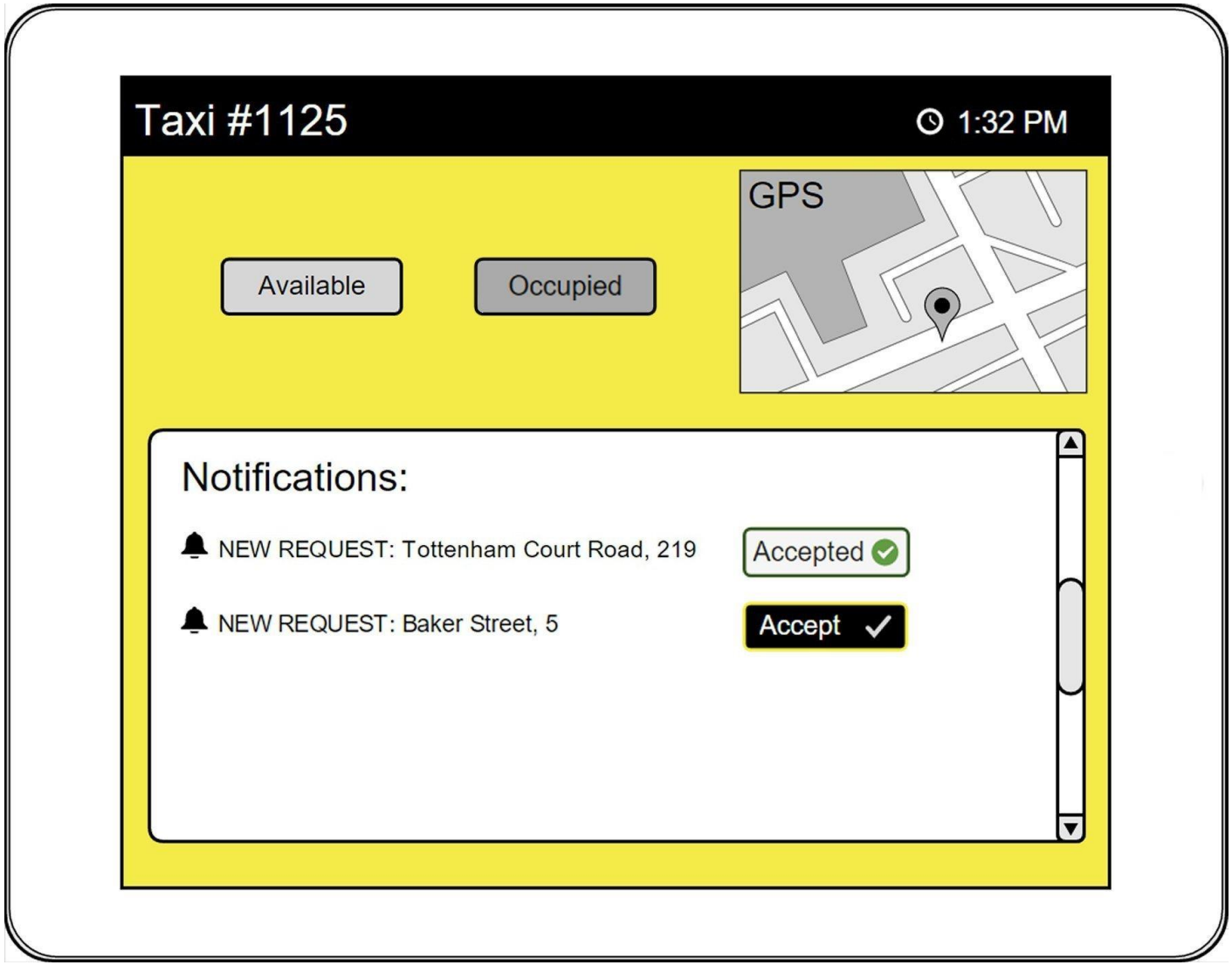

My Taxi Service
http://mytaxiservice.com/john-jones/information
My Taxi Service
John | Personal Info | Your Reservations | Logout
Name:
John
Surname:
Jones
E-mail:
john.jones@gmail.com
Edit
Mobile Phone:
+4422550099
Edit
2/2
Credit Card:
MasterCard, * * * *2323
Expiration Date:
09/17
Edit
Delete
Add
Username
Jhonny46
Edit
Password:
****
Edit
Save changes
HOMEPAGE | Info

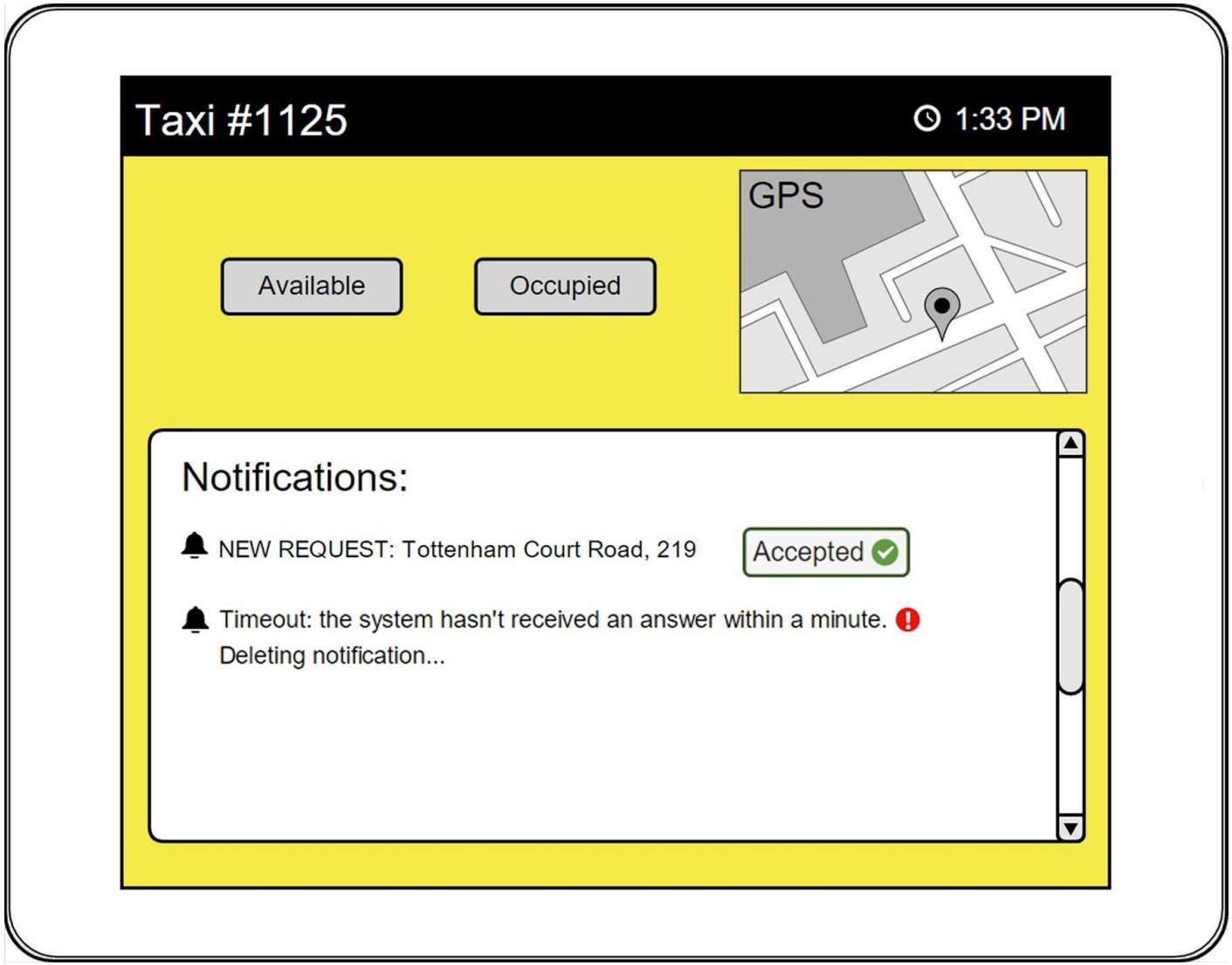
And now, here are some mockups of the taxi driver application installed on board:

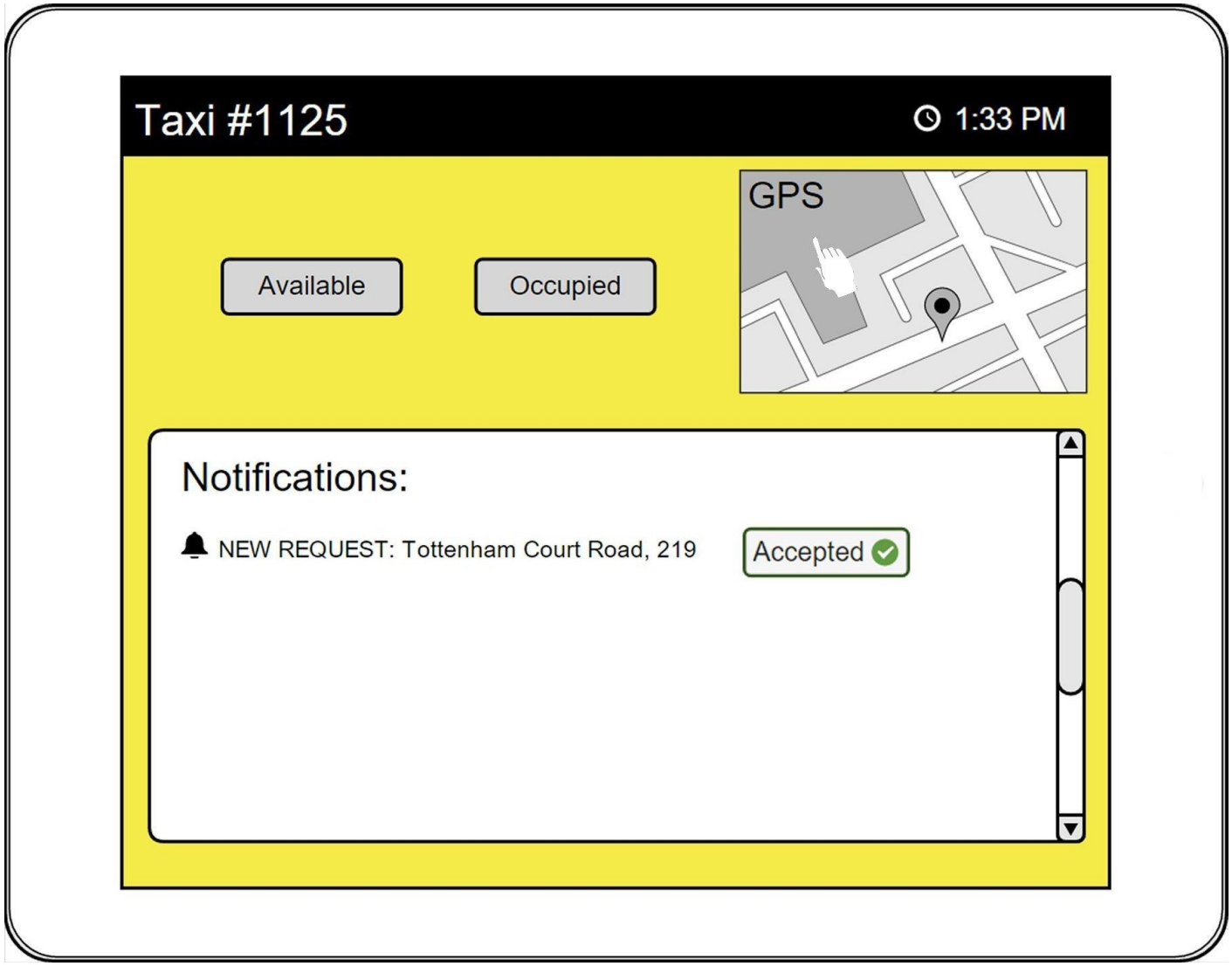


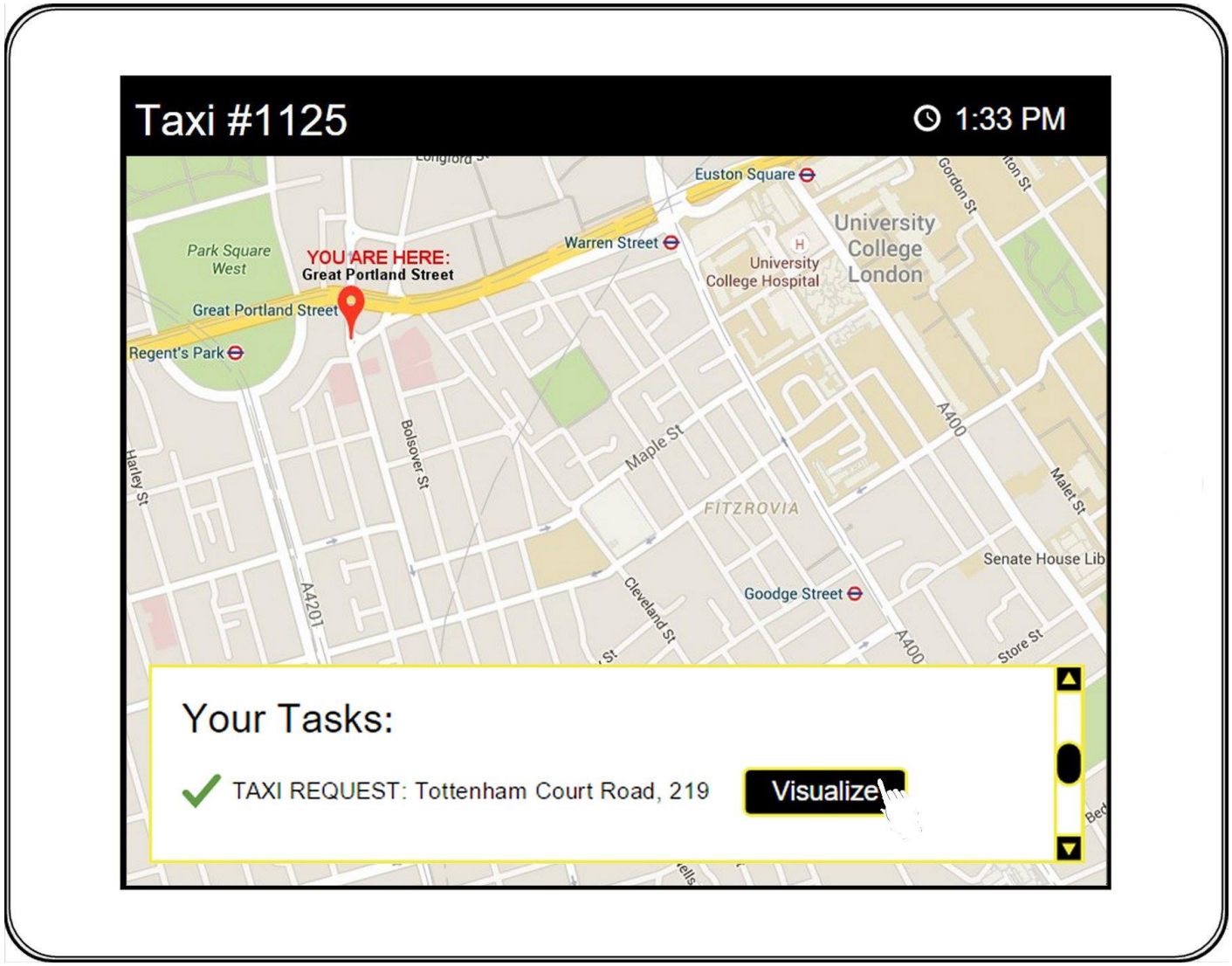

Taxi #1125
1:31 PM
GPS
Available
Occupied
Notifications:
NEW REQUEST: Tottenham Court Road, 219
Accept

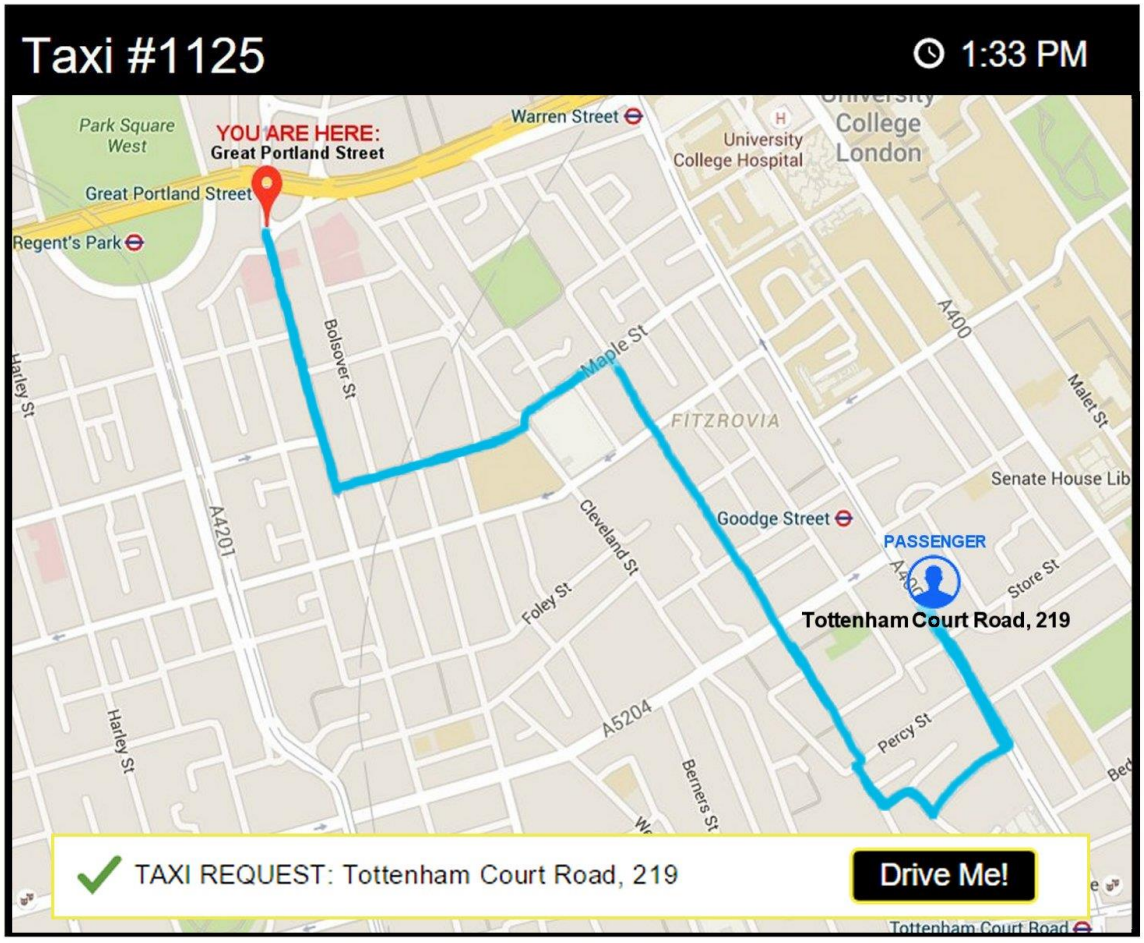

Taxi #1125
1:31 PM
GPS
Available
Occupied
Notifications:
NEW REQUEST: Tottenham Court Road, 219
Accepted


Taxi #1125
1:32 PM
GPS
Available
Occupied
Notifications:
NEW REQUEST: Tottenham Court Road, 219
Accepted
NEW REQUEST: Baker Street, 5
Accept


Taxi #1125
1:33 PM
GPS
Available
Occupied
Notifications:
NEW REQUEST: Tottenham Court Road, 219
Accepted
Timeout: the system hasn't received an answer within a minute.
Deleting notification...


Taxi #1125
1:33 PM
GPS
Available
Occupied
Notifications:
NEW REQUEST: Tottenham Court Road, 219
Accepted


Taxi #1125
1:33 PM
Euston Square
Gordon St
University College London
University College Hospital
Warren Street
YOU ARE HERE:
Great Portland Street
Park Square West
Great Portland Street
Regent's Park
Bolsover St
Maple St
FITZROVIA
A400
Malet St
Senate House Lib
Goodge Street
Cleveland St
A4201
Store St
Harley St
Your Tasks:
TAXI REQUEST: Tottenham Court Road, 219
Visualize


Taxi #1125
1:33 PM
Park Square West
YOU ARE HERE:
Great Portland Street
Warren Street
Great Portland Street
Regent's Park
University College Hospital
College London
Bolsover St
Maple St
Harley St
A400
Malet St
FITZROVIA
Senate House Lib
Goodge Street
PASSENGER
Cleveland St
A4201
Foley St
Store St
Tottenham Court Road, 219
A5204
Percy St
Harley St
Berners St
TAXI REQUEST: Tottenham Court Road, 219
Drive Me!
Tottenham Court Road

5 Requirements Traceability

In this last paragraph of our Design Document, we describe the correspondences between the requirements we have identified in the RASD document and the components characterizing our system, described earlier in the document.

In other words, we are going to describe and explain how our requirements map into the design elements previously identified.

First, we report here the general requirements described in section 4 of our RASD document, highlighting the components (sub-systems) that are involved in their satisfiability:

Registration:

- The **Sign Up System** should provide a “sign up” functionality: exploiting this functionality the guest can register to the application, filling the related form. The **Sign Up System** has to verify the inputs inserted: if the user already exists or the data are wrong, this is communicated to the **Guest Manager** that shows an error message to the guest.
- The **Sign Up System** should create a new profile in the passengers database and the **Log In System** should allow the new user to log in.
  The passengers database is managed by the **Data System**.
- The **Data System** has to store user information including name, surname, e-mail, mobile phone number, credit card, username and password.

Log In:

- The **Log In System** manages the “log in” functionality that allows the user to access the application and grants the access to the application only after a user types a correct username and password. It also communicates with the **Guest Manager** that eventually shows an error message whenever the authentication is wrong.
- When the login is corrected, the **User System** should show the user his personal page.

Insert, edit, delete personal information:

- The **Profile Managing System** allows the user to insert new personal data inside his profile information, to modify and to delete existing personal information.

Request a taxi:

- The **Request System** provide a “request” functionality, allowing a passenger to request a taxi from 6:00 a.m. to 9:00 p.m. The **Request System** receive the input inserted by the user, verifies them and, in case of successful request, creates a new request that has to be stored in the database.
- The **Driver Communication System** has to forward each request to the first taxi situated in the requested zone, in order to check its availability, and then mobilize the first available taxi in the queue of that zone.
- The **Queue Manager** should move a taxi to the bottom of the queue if not available for one of the current requests.

- After a taxi is associated with a specific request and it's mobilized, the **User Notification System** sends a confirmation notification (both by e-mail and showing it in his notifications box), communicating the code of the taxi and the waiting time.

Reserve a taxi:

- The **Reservation System** should provide a "reserve" functionality in order to allow passenger to reserve a taxi (from 6:00 a.m. to 9:00 p.m.) for a specific ride.
  The **Reservation System** receive the input inserted by the user, verifies them and, in case of successful request, creates a new request that has to be stored in the database.
- The **User Notification System** has to confirms the reservation to the passenger, if it is received at least 2 hours before the ride.
- The **User System** should show an error message when the reservation is not done at least 2 hours in advanced.
- The **Queue Manager** should allocate for every reservation one of the available taxis 10 minutes before the established meeting time.
- The **Profile Managing System** should provide a function that allows a user to visualize the list of reservation he has done.

Drivers communication

- The **Driver Communication System** should be able to receive communication and information about availability from each taxi driver.
- The **Taxi Manager** should store all the information about each taxi's availability.
- When a request from the system is received, a taxi driver should communicate its availability to the **Driver Communication System**.
- When a taxi driver decides to take care of certain calls, he should inform the **Driver Communication System**.
- The **Driver Communication System** should send information about extraordinary maintenance of the site to all the taxi drivers.
- The **Driver Communication System** should send a notification to all the taxi drivers saying that the web service is active or inactive.

Localization

- The **Taxi Manager** is able to accurately localize the distribution of the taxis in each zone using a GPS locator and a specific function.

Notifications:

- The **User Notification System** has to notify a user, confirming his request, when a taxi is available.
- After confirming a request, the **User Notification System** should communicate to the passenger the code of the incoming taxi and the estimated waiting time.

- The **User Notification System** should confirm a successful reservation sending a notification to the passenger.

Payments:

- The **Payments Manager** checks the payments of all the passengers every month.
    - The **User Notification System** should send an e-mail notification to inform all those users who doesn't pay the taxi requested or reserved, or who doesn't show up, that the payment has not been accomplished; they will have to pay a penalty fee within 2 month.
- The **User Notification System** should send an e-mail to all those users who has already paid, confirming the payments.

We report now a table containing a summary of the requirements-components mapping (see the following page):

		Design Component												
		GuestManager		RegisteredUserManager					TaxiManager			PaymentsManager	Database	
			Sign Up System	Login System	UserSystem					DriverCommunicationSystem	QueueManager		BackUpSystem	DataSystem
					ProfileManagingSystem	RequestSystem	ReservationSystem	UserNotificationSystem						
Requirement	1. The guest can sign up.		x										x	x
	2. The guest visualizes an error message when the authentication is wrong.	x		x										
	3. The registered user can log in.			x										
	4. The registered user can visualize his personal page.				x									
	5. The registered user can modify his personal information.				x								x	x
	6. The registered user can delete his personal information.				x								x	x
	7. The registered user can request a taxi filling out the related form.					x							x	x
	8. The registered user can reserve a taxi filling out the related form.						x						x	x
	9. The registered user can receive a taxi request confirmation.							x						
	10. The registered user can receive a taxi reservation confirmation.							x						
	11. The registered user can receive information about the code and the wainting time of the incoming taxi.							x						
	12. The registered user can visualize his previous and current reservations and decide to delete one of them.				x		x							
	13. The registered user visualizes an error message when the reservation is not done at least 2 hours in advanced.				x		x							
	14. The system should allocate for every reservation one of the available taxis 10 minutes before the established meeting time.						x			x	x		x	x
	15. The system has to forward each request to the first taxi situated in the requested zone, in order to check its availability, and then mobilize the first available taxi in the queue of that zone.									x	x			
	16. The system should move a taxi to the bottom of the queue if not available for one of the current requests.										x		x	x
	17. The system should store all the information about each taxi's availability.												x	x
	18. The system takes care of being able to accurately localize the distribution of the taxis in each zone using a GPS locator.								x					
	19. The system checks the payments of all the passengers every month.											x		
	20. The system should send an e-mail notification to inform all those users who doesn't pay the taxi requested or reserved, or who doesn't show up, that the payment has not been accomplished; they will have to pay a penalty fee within 2 month.							x				x		
	21. The system should send information about extraordinary maintenance of the site to all the taxi drivers.									x				
	22. The system should send an e-mail to all those users who has already paid, confirming the payments.							x				x		
	23. The system should send a notification to all the taxi drivers saying that the system is active or not active.									x				
	24. The taxi driver can send information about his availability.									x				
	25. The taxi driver can send information about the calls he takes care of.									x				
	26. The taxi driver can receive information about the requests and the reservations he has to satisfy.									x				

6 Used Tools

In this paragraph we are going to list all the tools we have used to create this document:

- *Microsoft Office Word 2010* : to redact and format this document
- *Microsoft Office Excel 2013* : to create the requirements traceability table
- *Moqups + a photo editor*: to create mockups
- *Draw.io* : to create UML Sequence Diagrams, Component and Deployment Diagrams
- *Dia* : to create architecture schemas and ER Diagram

7 Working Hours

In order to redact and write this document we spent 25 hours per person.

8 Appendixes

8.1 RASD modifications

We had to add some adjustments in chapter 4 “Requirements” of our RASD document.

In particular, in the requirements section concerning “Drivers communication”, we need some new requirements:

- when the system becomes inactive, at 9:00 p.m., the system sends a notification to all the taxi drivers to inform them that the web service is not active;
- when the system resumes its activity, at 6:00 a.m., the system notified all the taxi drivers communicating that the web service is now active;
- the maintenance of the site is performed when the system is not active. However, if a situation of extraordinary maintenance of the application occurs, the taxi drivers are notified with a “down system notification”.