AUBS Lines (American University in Bulgaria Shinkansen Lines)

SPRINT 1 Deliverables

AUBS LINES 10/9/16 COS 315 – Software Engineering

CONTENT

Product Backlog and Sizes

Planning Poker - Planning poker was played online via PlanningPoker.com website. Most of the stories were easy to agree on, but however few of them required further discussion to resolve some conflicts between the three of us in terms of evaluation.

Product Backlog List (This list is subject to changes if and when new user stories are added)

User Story	Tasks Required	Task Pts	Total Points	Priority
As a prospective passenger, I want to be able to register with the system.	Implement Register/Login from the home page	1	11	High
	Create a register form including username, password, phone, e-mail.	3		
	Login Form with username/password	2		
	Database for saving records	3		
	Validation of user password and credentials	2		
As a registered passenger, I want to be able to book seats, i.e. buy tickets.	Querying the trip database for selecting relevant trips	2	10	High
	Confirmation Page of Booking	5		
	Availability of Seats Validation	1		
	Update Trip Database	2		
As a registered passenger with a booked train journey, I want to be able to reserve seat(s) for my journey.	Reserve a specific seat in the booking	1	11	Medium
	Bulk Reservation in the booking	2		
	Get Available Seats for specific journeys	2		
	Validate availability of specific seats	3		
	Revise the functionality of updating the trip database to account for specific seats.	3		
As an administrator, I want to be able to enter new train journey details, amend existing journey details, and delete existing journey details.	Administrator Validation in the login form	1	8	High
	Get all Trips from database	2		
	Edit View of Available trips to add admin functions (Create, Edit, Delete)	2		
	Update trip database after creation	3		
As a booked passenger, I want to be able to cancel my journey for personal reasons, and receive a refund.	Create View with booked trips for every user	2	7	Medium
	Get user trips from database	1		
	Update seat availability after Cancelation	3		
	Update user's balance after cancelation	1		

As an administrator, I want to be able to cancel a train journey for operational reasons, and give a refund to booked customers.	Query the trips to obtain all affected customers	1	4	Low
	Cancel Trips in the Admin View	1		
	Update the balance of affected customers	2		

Sprint Backlog

Deliverable: User Stories that were selected by the team to be implemented in this sprint.

User Story	Tasks Required	Actual Task Pts	Total Points	Initial Estimation
As a prospective passenger, I want to be able to register with the system.	Implement Register/Login from the home page	4	15	11
	Create a register form including username, password, phone, e-mail.	3		
	Login Form with username/password	2		
	Database for saving records	2		
	Validation of user password and credentials	4		
As a registered passenger, I want to be able to book seats, i.e. buy tickets.	Querying the trip database for selecting relevant trips	5	15	10
	Confirmation Page of Booking	5		
	Availability of Seats Validation	3		
	Update Trip Database after Booking	1		
As an administrator, I want to be able to enter new train journey details, amend existing journey details, and delete existing journey details.	Administrator Validation in the login form	3	10	8
	Get all Trips from database	2		
	Edit View of Available trips to add admin functions (Create, Edit, Delete)	2		
	Update trip database after creation of trip	3		

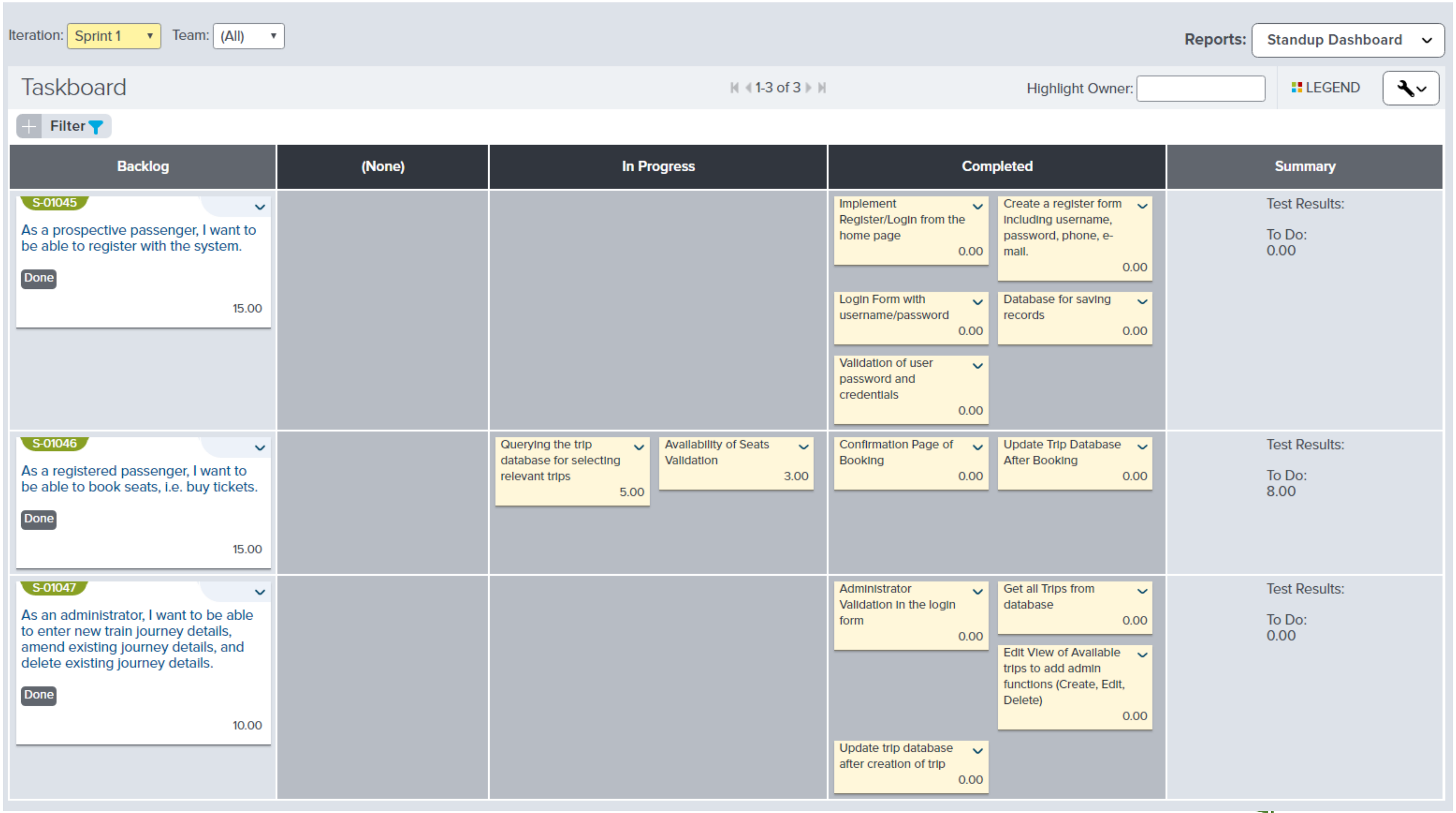
Scrum Board Maintenance

Deliverable: Photos of Scrum meetings, Task Board over the sprint

The purpose of the Scrum Board (Task Board) is to maintain a progress indicator over the sprint showing which stage each user story is at. This, is also manageable by a burnt down chart, that shows in addition to the task board, how the speed of the team has been over the sprint period, determining the distribution of points (working hours) per day.

Team Photos

Task Board for SPRINT 1



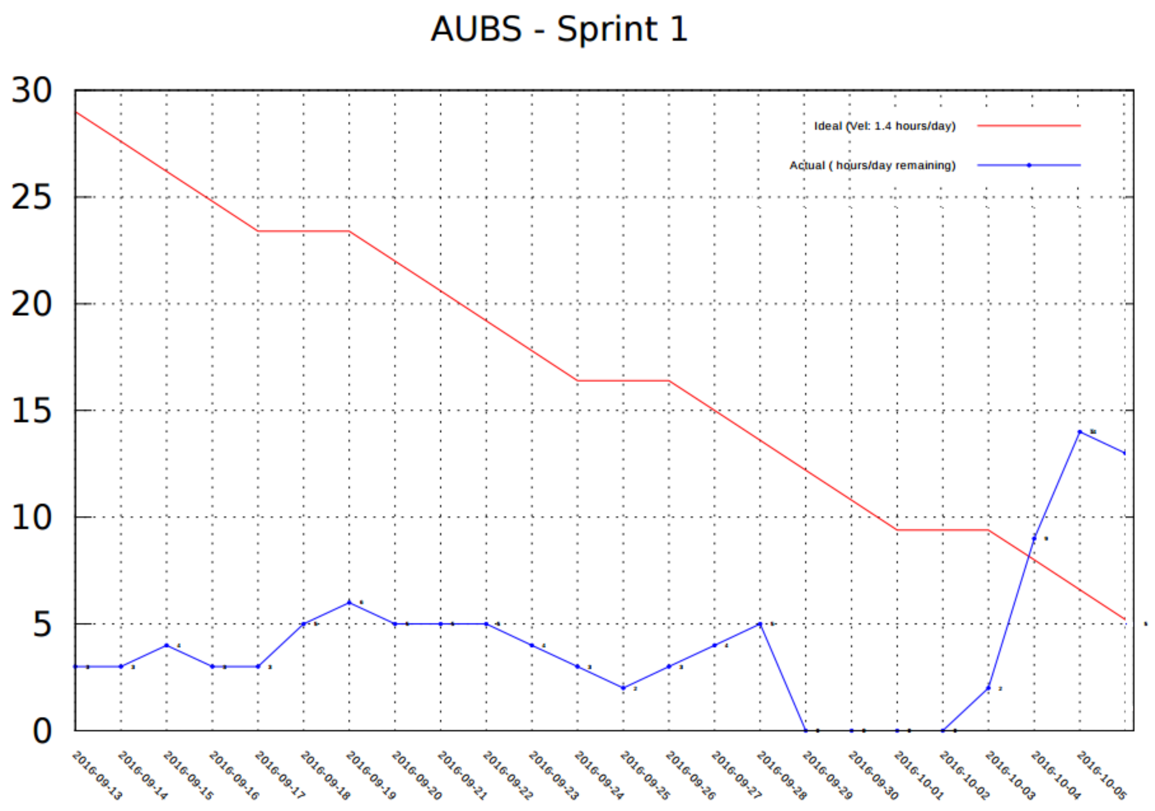
Iteration: Sprint 1 Team: (All) Reports: Standup Dashboard

Taskboard 1-3 of 3 Highlight Owner: LEGEND

Filter

Backlog	(None)	In Progress	Completed	Summary
S-01045 As a prospective passenger, I want to be able to register with the system. Done 15.00			Implement Register/Login from the home page 0.00; Create a register form including username, password, phone, e-mail. 0.00; Login Form with username/password 0.00; Database for saving records 0.00; Validation of user password and credentials 0.00	Test Results: To Do: 0.00
S-01046 As a registered passenger, I want to be able to book seats, i.e. buy tickets. Done 15.00		Querying the trip database for selecting relevant trips 5.00; Availability of Seats Validation 3.00	Confirmation Page of Booking 0.00; Update Trip Database After Booking 0.00	Test Results: To Do: 8.00
S-01047 As an administrator, I want to be able to enter new train journey details, amend existing journey details, and delete existing journey details. Done 10.00			Administrator Validation in the login form 0.00; Get all Trips from database 0.00; Edit View of Available trips to add admin functions (Create, Edit, Delete) 0.00; Update trip database after creation of trip 0.00	Test Results: To Do: 0.00

Deliverable: Photo of the burn-down chart for the sprint

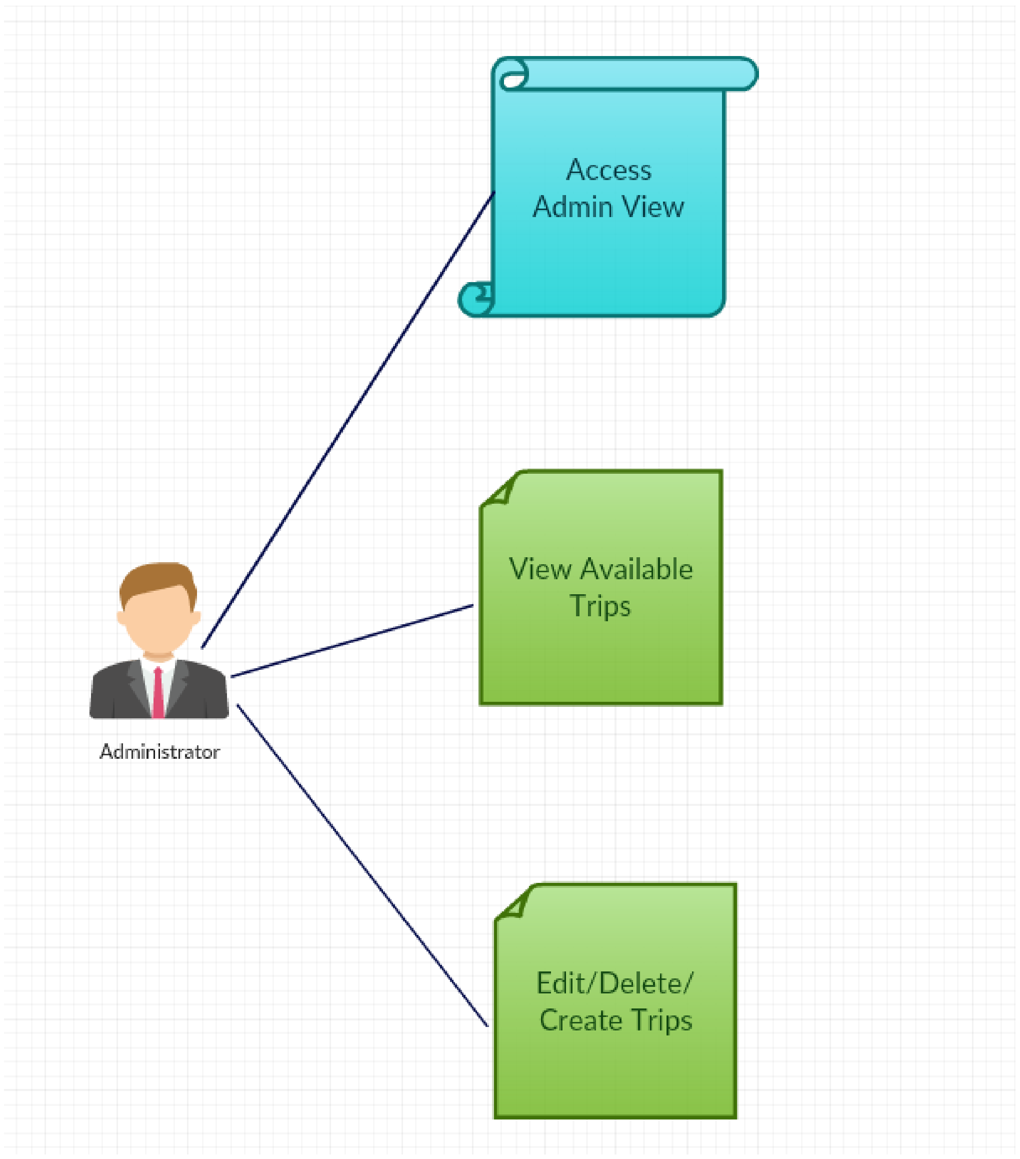


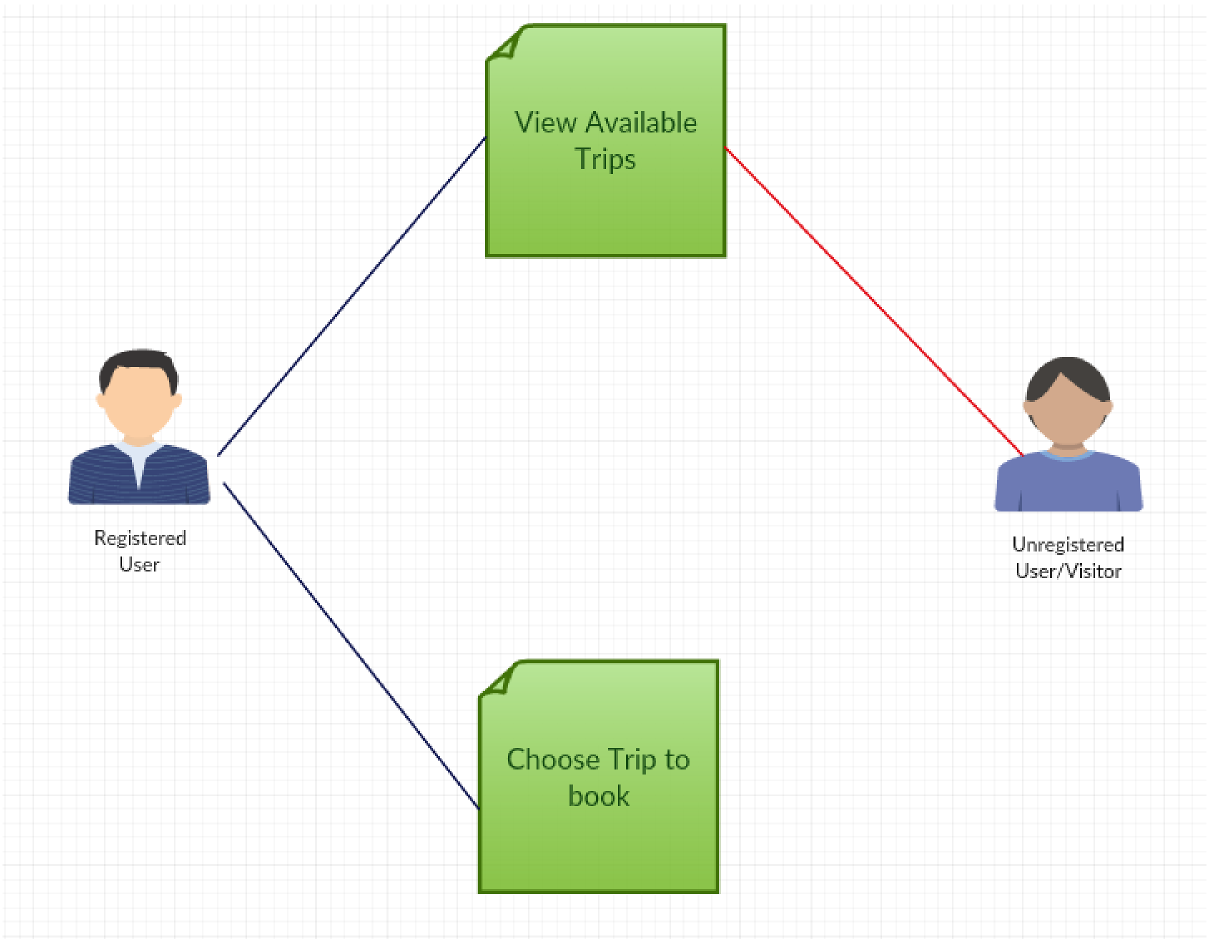
Working hours here are interchangeable with points, in order to measure velocity.

I.E. 1 working hour = 1 point

User Case Diagrams

Administrator Login / Edit





Registerd/ Unregistered User Case

Code:

Below the basic code for models/controllers is attached. The project is built on ASP.NET and C#. Up to now, there are only 3 MVC structures, which is Users, Trips and Bookings. For all of them there coincides a table in our database, while the last one(Bookings) is a relation between the two first.

User(Account Model):

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace TrainReservation.Models
{
    public class ExternalLoginConfirmationViewModel
    {
        [Required]
        [Display(Name = "Email")]
        public string Email { get; set; }
    }

    public class ExternalLoginListViewModel
    {
        public string ReturnUrl { get; set; }
    }

    public class SendCodeViewModel
    {
        public string SelectedProvider { get; set; }
        public ICollection<System.Web.Mvc.SelectListItem> Providers { get; set; }
        public string ReturnUrl { get; set; }
        public bool RememberMe { get; set; }
    }

    public class VerifyCodeViewModel
    {
        [Required]
        public string Provider { get; set; }

        [Required]
        [Display(Name = "Code")]
        public string Code { get; set; }
        public string ReturnUrl { get; set; }

        [Display(Name = "Remember this browser?")]
        public bool RememberBrowser { get; set; }

        public bool RememberMe { get; set; }
    }

    public class ForgotViewModel
    {
        [Required]
        [Display(Name = "Email")]
        public string Email { get; set; }
    }

    public class LoginViewModel
    {
        [Required]
        [Display(Name = "Email")]
        [EmailAddress]
        public string Email { get; set; }

        [Required]
        [DataType(DataType.Password)]
        [Display(Name = "Password")]
        public string Password { get; set; }

        [Display(Name = "Remember me?")]
        public bool RememberMe { get; set; }
    }

    public class RegisterViewModel
    {
        [Required]
        [EmailAddress]
        [Display(Name = "Email")]
        public string Email { get; set; }

        [Required]
        [StringLength(100, ErrorMessage = "The {0} must be at least {2} characters 
long.", MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "Password")]
        public string Password { get; set; }

        [DataType(DataType.Password)]
        [Display(Name = "Confirm password")]
        [Compare("Password", ErrorMessage = "The password and confirmation password do 
not match.")]
        public string ConfirmPassword { get; set; }
    }

    public class ResetPasswordViewModel
    {
        [Required]
        [EmailAddress]
        [Display(Name = "Email")]
        public string Email { get; set; }

        [Required]
        [StringLength(100, ErrorMessage = "The {0} must be at least {2} characters 
long.", MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "Password")]
        public string Password { get; set; }

        [DataType(DataType.Password)]
        [Display(Name = "Confirm password")]
        [Compare("Password", ErrorMessage = "The password and confirmation password do 
not match.")]
```

```
        public string ConfirmPassword { get; set; }

        public string Code { get; set; }
    }

    public class ForgotPasswordViewModel
    {
        [Required]
        [EmailAddress]
        [Display(Name = "Email")]
        public string Email { get; set; }
    }
}
```

User(Account) Controller:

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Dynamic;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using TrainReservation.Models;
using TrainReservationDBContext;

namespace TrainReservation.Controllers
{
    public class TripsController : Controller
    {
        private TrainReservationDbContext db = new TrainReservationDbContext();

        // GET: Trips
        public ActionResult Index()
        {
            return View(db.Trips.ToList());
        }

        // GET: Trips/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Trip trip = db.Trips.Find(id);
            if (trip == null)
            {
                return HttpNotFound();
            }
            return View(trip);
        }

        // GET: Trips/Create
        public ActionResult Create()
        {
            return View();
        }

        // POST: Trips/Create
        // To protect from overposting attacks, please enable the specific properties 
you want to bind to, for 
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create([Bind(Include = 
"TripID,Departure,Departure_Time,Destination,Arrival_Time,Seats,Price")] Trip trip)
        {
            if (ModelState.IsValid)
            {
                db.Trips.Add(trip);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            return View(trip);
        }

        // GET: Trips/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Trip trip = db.Trips.Find(id);
            if (trip == null)
            {
                return HttpNotFound();
            }
            return View(trip);
        }



        // POST: Trips/Edit/5
        // To protect from overposting attacks, please enable the specific properties 
you want to bind to, for 
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include = 
"TripID,Departure,Departure_Time,Destination,Arrival_Time,Seats,Price")] Trip trip)
        {
            if (ModelState.IsValid)
            {
                db.Entry(trip).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            return View(trip);
        }

        // GET: Trips/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Trip trip = db.Trips.Find(id);
            if (trip == null)
            {
                return HttpNotFound();
            }
            return View(trip);
        }

        // POST: Trips/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Trip trip = db.Trips.Find(id);
            db.Trips.Remove(trip);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        public ActionResult Book(int? id, string sender)
        {
            Bookings booking = new Bookings();

            booking.TripID = (int)id;
            booking.UserId = sender;

            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Trip trip = db.Trips.Find(id);
            if (trip == null)
            {
                return HttpNotFound();
            }

            BookingConfirmationModel mymodel = new BookingConfirmationModel();

            mymodel.Trips = trip;
            mymodel.Bookings = booking;



            return View(mymodel);

        }
        // POST: Trips/Delete/5
        [HttpPost, ActionName("Book")]
        [ValidateAntiForgeryToken]
        public ActionResult BookingConfirmed(int?id, string sender)
        {



            Bookings booking = new Bookings();
            booking.TripID = (int)id;
            booking.UserId = sender;
            db.Bookings.Add(booking);
            db.Trips.Find(booking.TripID).Seats--;
            db.SaveChanges();
            return RedirectToAction("Confirmed");
        }

        public ActionResult Confirmed()
        {
            return View();

        }

        private dynamic GetTrips()
        {
            throw new NotImplementedException();
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}
```

Trips Model:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Data.Entity;
using System.ComponentModel.DataAnnotations;
using TrainReservationDBContext;

namespace TrainReservation.Models
{
    public class Trip
    {
        public int TripID { get; set; }
        public string Departure { get; set; }
       [ Display(Name = "Departure Time")]
        public DateTime Departure_Time { get; set; }
        public string Destination { get; set; }
        [Display(Name = "Arrival Time")]
        public DateTime Arrival_Time { get; set; }
        public int Seats { get; set; }
        public decimal Price { get; set; }

    }


}
```

```
Trips Controller:

using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Dynamic;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using TrainReservation.Models;
using TrainReservationDBContext;

namespace TrainReservation.Controllers
{
    public class TripsController : Controller
    {
        private TrainReservationDbContext db = new TrainReservationDbContext();

        // GET: Trips
        public ActionResult Index()
        {
            return View(db.Trips.ToList());
        }

        // GET: Trips/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Trip trip = db.Trips.Find(id);
            if (trip == null)
            {
                return HttpNotFound();
            }
            return View(trip);
        }

        // GET: Trips/Create
        public ActionResult Create()
        {
            return View();
        }

        // POST: Trips/Create
        // To protect from overposting attacks, please enable the specific properties 
you want to bind to, for 
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create([Bind(Include = 
"TripID,Departure,Departure_Time,Destination,Arrival_Time,Seats,Price")] Trip trip)
        {
            if (ModelState.IsValid)
            {
                db.Trips.Add(trip);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            return View(trip);
        }

        // GET: Trips/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Trip trip = db.Trips.Find(id);
            if (trip == null)
            {
                return HttpNotFound();
            }
            return View(trip);
        }
        // POST: Trips/Edit/5
        // To protect from overposting attacks, please enable the specific properties
you want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include =
"TripID,Departure,Departure_Time,Destination,Arrival_Time,Seats,Price")] Trip trip)
        {
            if (ModelState.IsValid)
            {
                db.Entry(trip).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            return View(trip);
        }

        // GET: Trips/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Trip trip = db.Trips.Find(id);
            if (trip == null)
            {
                return HttpNotFound();
            }
            return View(trip);
        }

        // POST: Trips/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Trip trip = db.Trips.Find(id);
            db.Trips.Remove(trip);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        public ActionResult Book(int? id, string sender)
        {
            Bookings booking = new Bookings();

            booking.TripID = (int)id;
            booking.UserId = sender;

            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Trip trip = db.Trips.Find(id);
            if (trip == null)
            {
                return HttpNotFound();
            }

            BookingConfirmationModel mymodel = new BookingConfirmationModel();

            mymodel.Trips = trip;
            mymodel.Bookings = booking;




            return View(mymodel);

        }

        // POST: Trips/Delete/5
        [HttpPost, ActionName("Book")]
        [ValidateAntiForgeryToken]
        public ActionResult BookingConfirmed(int?id, string sender)
        {



            Bookings booking = new Bookings();
            booking.TripID = (int)id;
            booking.UserId = sender;
            db.Bookings.Add(booking);
            db.Trips.Find(booking.TripID).Seats--;
            db.SaveChanges();
            return RedirectToAction("Confirmed");
        }

        public ActionResult Confirmed()
        {
            return View();

        }

        private dynamic GetTrips()
        {
            throw new NotImplementedException();
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}
```

Testing:

Deliverable: Test plan for acceptance testing, acceptance tests for each user story in the current sprint and results of every test, with an indication of pass or fail.

Our tests aim to cover: Usability, Functionality, and Performance.

Test 1: "Snappiness"

Type: Usability

Description:
We will measure by empirical statistics whether the average user find the application snappy at this stage".

For this test at least 10 testers should view and book trips and assess how well it performs (ranking from 1-10).

Expected Outcome: The testers should rate this with at least a 4.5 while the average of satisfaction should exceed 6.

-We cannot apply this measure yet to our application since it is on a very early stage yet with a lot of features missing.

Test 2: "Booking a Trip"

Users were asked to register, and then book a trip on the trips list. After this they were asked how did they find the overall experience and usability of the structure that is implemented up to present

The test was performed on 7 people who used the application on their own computers from a local host. The code for this test can be found above, in the code fragment attached for the trip controller

Results:

At least 50% rates the application > 4 on a scale from 1-10.

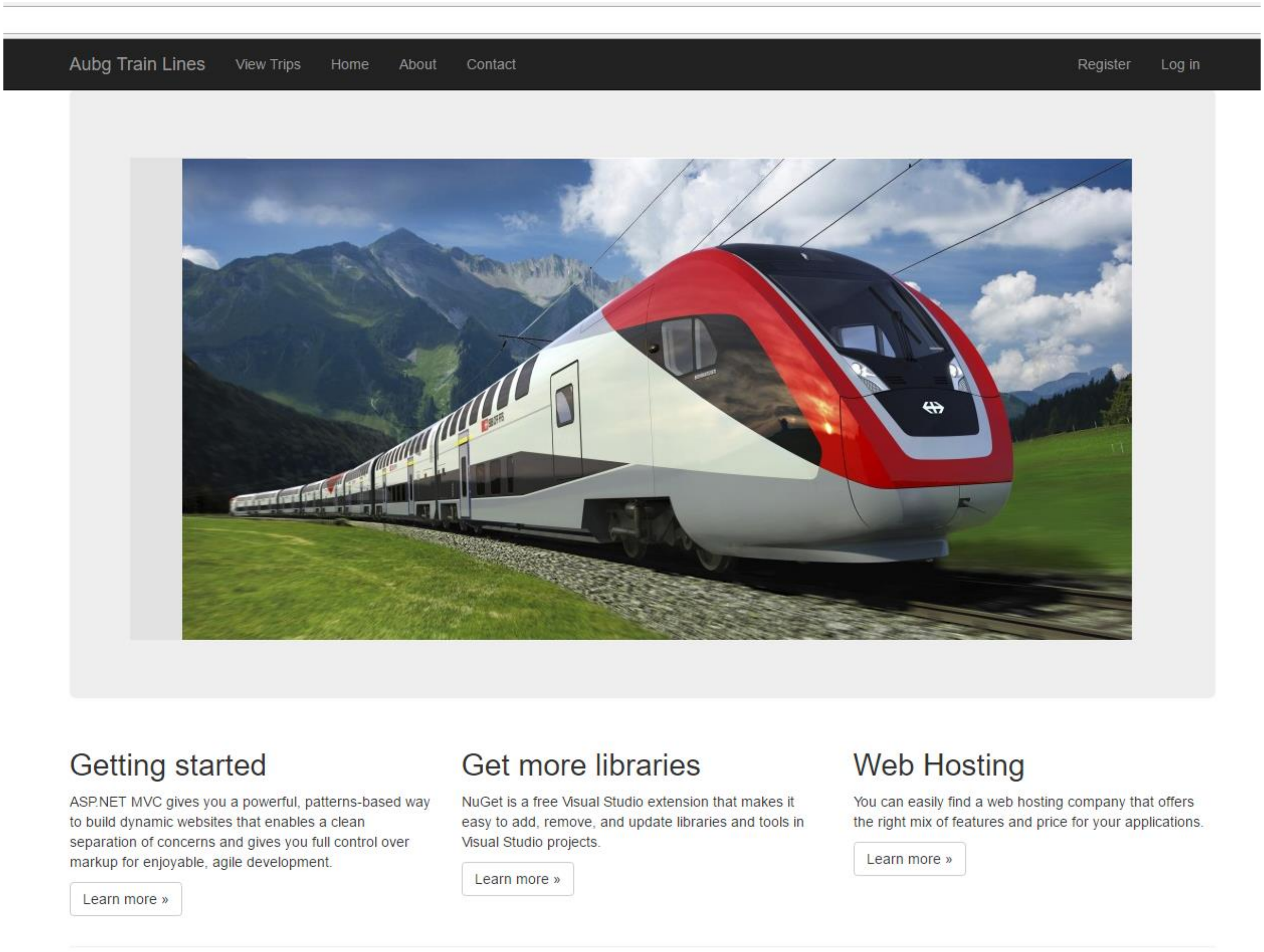
The low score must be result of the poor yet confirmation page that follows the booking.

Test 3: "Adding/Deleting or Editing a trip as Admin"

This test could not be performed to random users, since it requires the administrator of the train lines to try how it works. Hence, this test will be performed during the sprint review meeting where the product owner will be present.

ScreenShots of the Working Application up to the present stage:

Home Page



Getting started

ASP.NET MVC gives you a powerful, patterns-based way to build dynamic websites that enables a clean separation of concerns and gives you full control over markup for enjoyable, agile development.

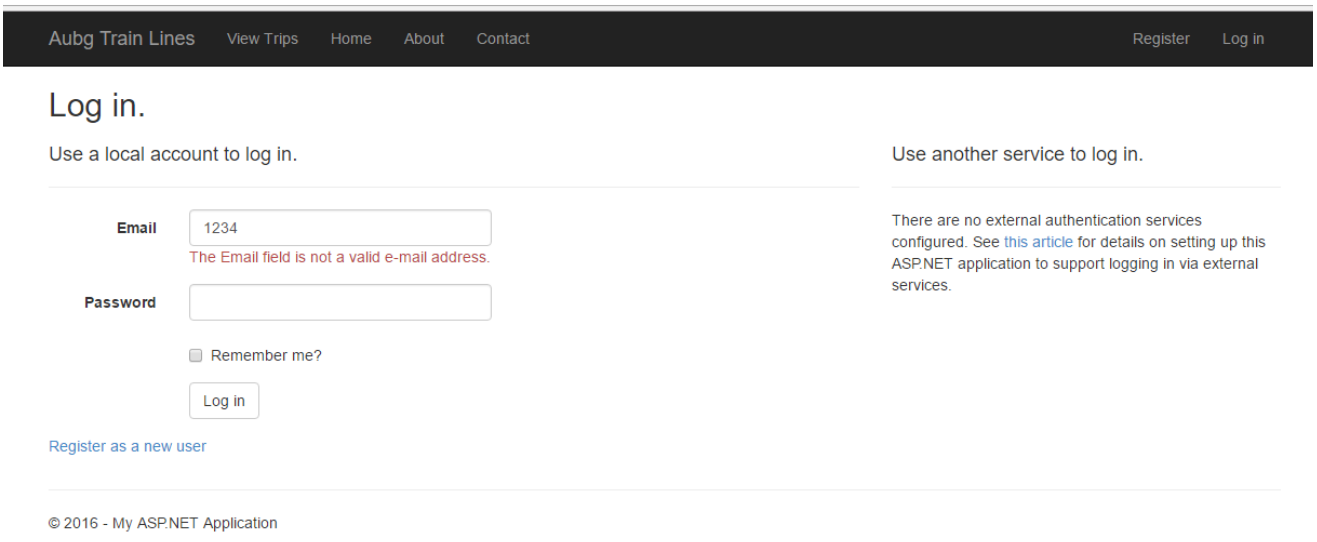
Get more libraries

NuGet is a free Visual Studio extension that makes it easy to add, remove, and update libraries and tools in Visual Studio projects.

Web Hosting

You can easily find a web hosting company that offers the right mix of features and price for your applications.

Log in



Log in.

Use a local account to log in.

Email: 1234

The Email field is not a valid e-mail address.

Password:

Remember me?

Log in

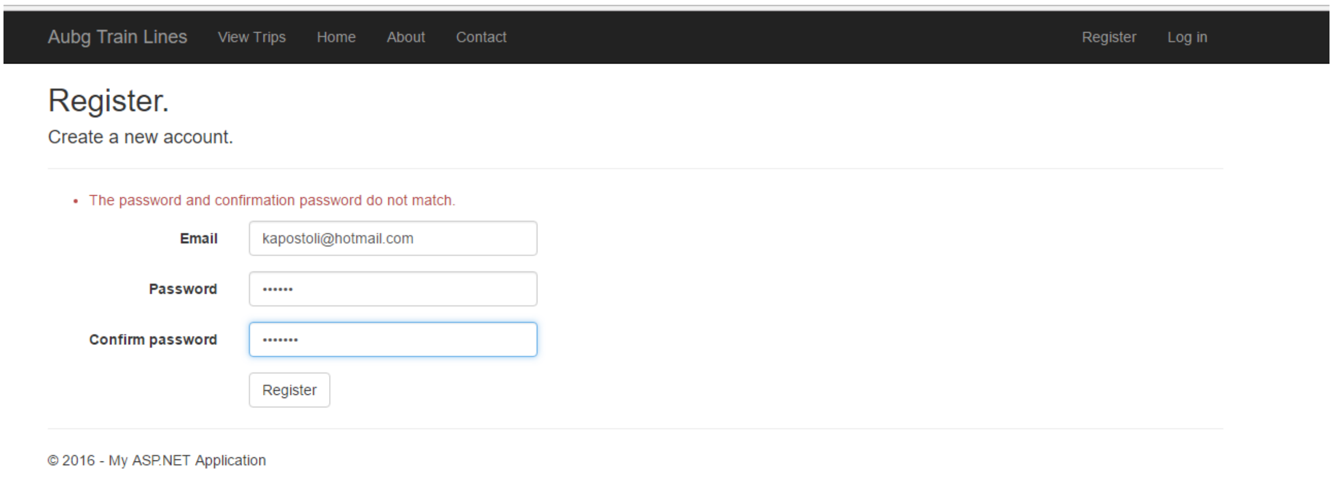
Register as a new user

Use another service to log in.

There are no external authentication services configured. See this article for details on setting up this ASP.NET application to support logging in via external services.

© 2016 - My ASP.NET Application

Register



Register.

Create a new account.

- The password and confirmation password do not match.

Email: kapostoli@hotmail.com

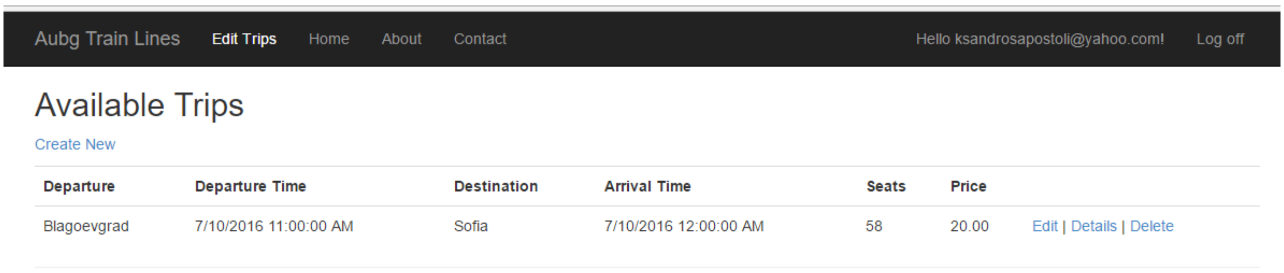
Password: ••••••

Confirm password: •••••••

Register

© 2016 - My ASP.NET Application

Edit Panel (Admin Panel)



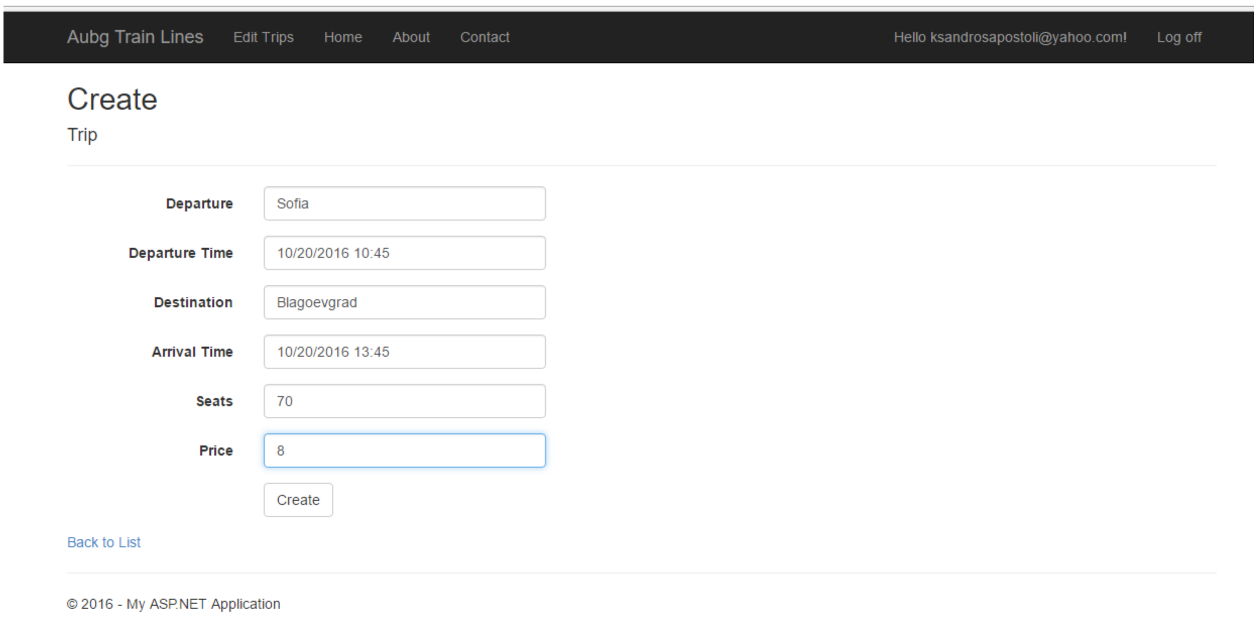
Available Trips

Create New

Departure	Departure Time	Destination	Arrival Time	Seats	Price	
Blagoevgrad	7/10/2016 11:00:00 AM	Sofia	7/10/2016 12:00:00 AM	58	20.00	Edit | Details | Delete

© 2016 - My ASP.NET Application

Create Trip View



Create

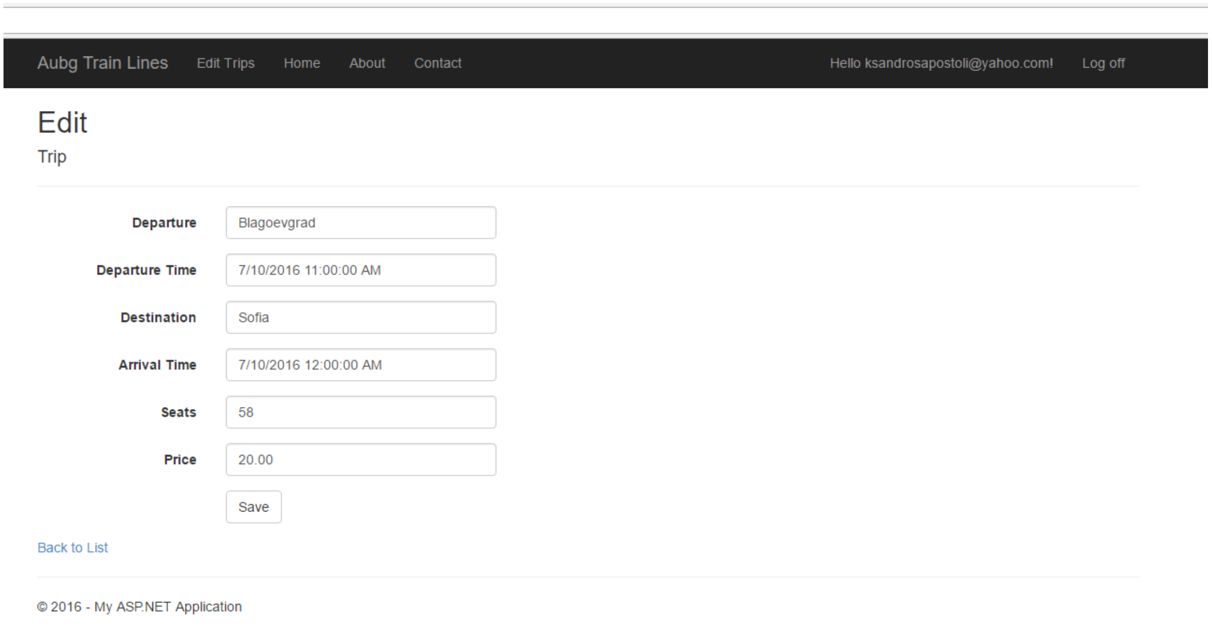
Trip

Departure	Sofia
Departure Time	10/20/2016 10:45
Destination	Blagoevgrad
Arrival Time	10/20/2016 13:45
Seats	70
Price	8

Back to List

© 2016 - My ASP.NET Application

Edit Trip (Authenticated as Admin only)



Aubg Train Lines Edit Trips Home About Contact Hello ksandrosapostoli@yahoo.com! Log off

Edit

Trip

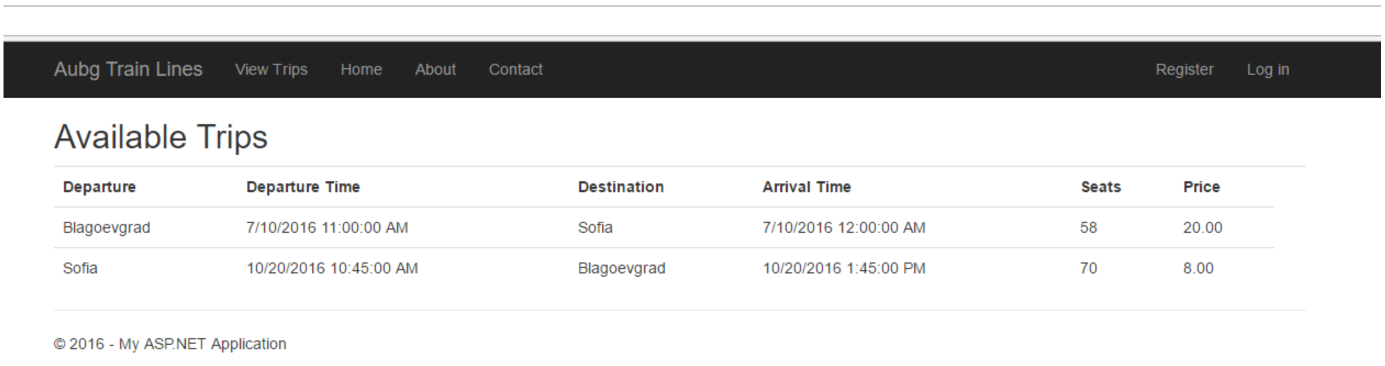
Departure	Blagoevgrad
Departure Time	7/10/2016 11:00:00 AM
Destination	Sofia
Arrival Time	7/10/2016 12:00:00 AM
Seats	58
Price	20.00

Save

Back to List

© 2016 - My ASP.NET Application

View Trips (As unregistered user)



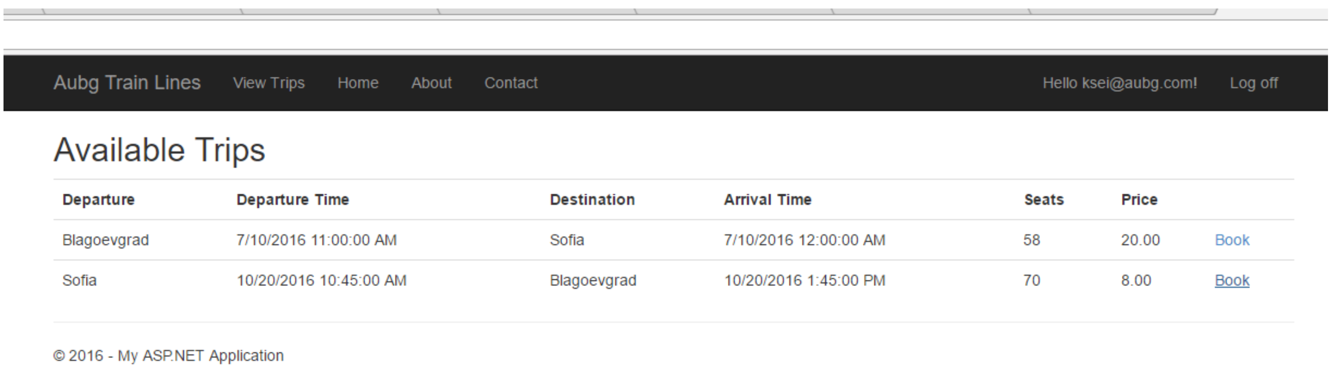
Aubg Train Lines View Trips Home About Contact Register Log in

Available Trips

Departure	Departure Time	Destination	Arrival Time	Seats	Price
Blagoevgrad	7/10/2016 11:00:00 AM	Sofia	7/10/2016 12:00:00 AM	58	20.00
Sofia	10/20/2016 10:45:00 AM	Blagoevgrad	10/20/2016 1:45:00 PM	70	8.00

© 2016 - My ASP.NET Application

View Trips (As authenticated User)

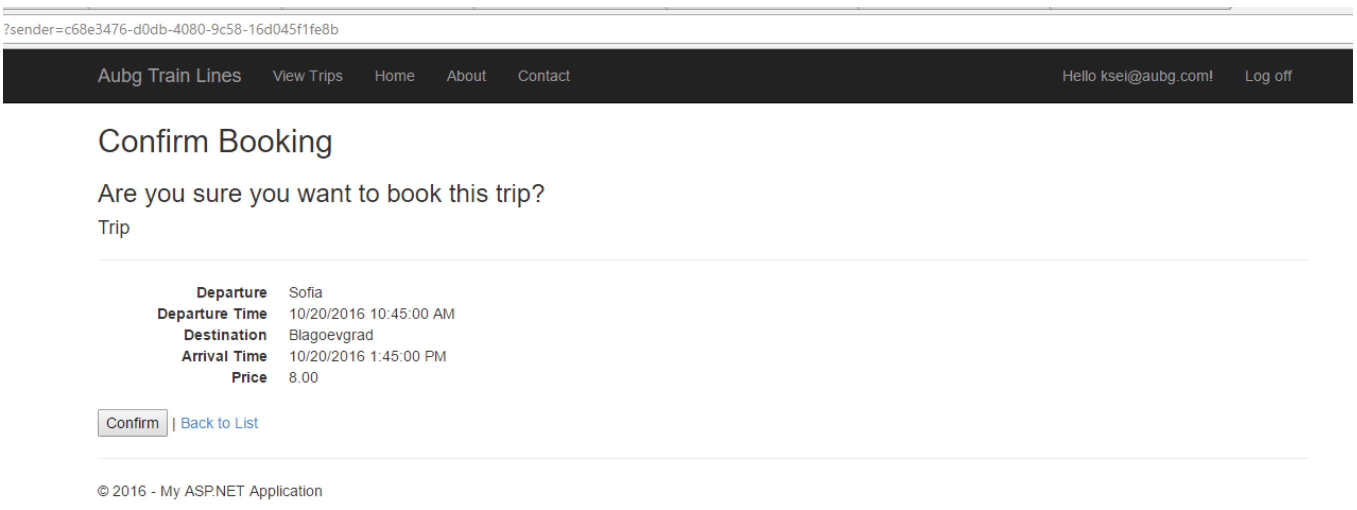


Available Trips

Departure	Departure Time	Destination	Arrival Time	Seats	Price	
Blagoevgrad	7/10/2016 11:00:00 AM	Sofia	7/10/2016 12:00:00 AM	58	20.00	Book
Sofia	10/20/2016 10:45:00 AM	Blagoevgrad	10/20/2016 1:45:00 PM	70	8.00	Book

© 2016 - My ASP.NET Application

Booking Confirmation Request



?sender=c68e3476-d0db-4080-9c58-16d045f1fe8b

Confirm Booking

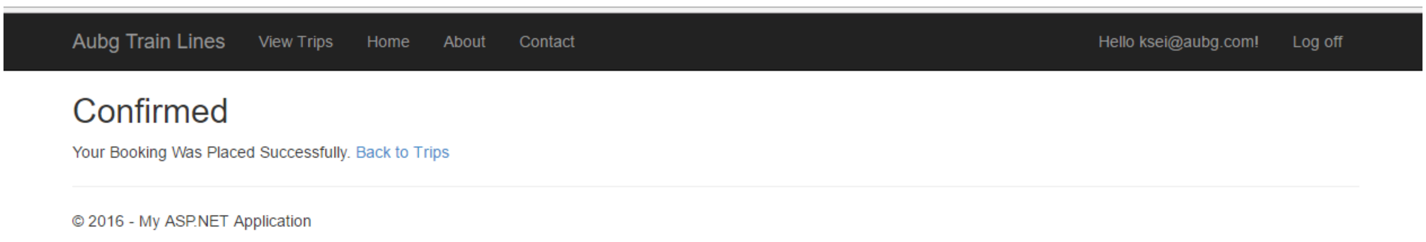
Are you sure you want to book this trip?

Trip

Departure	Sofia
Departure Time	10/20/2016 10:45:00 AM
Destination	Blagoevgrad
Arrival Time	10/20/2016 1:45:00 PM
Price	8.00

© 2016 - My ASP.NET Application

Booking Confirmation Page



Confirmed

Your Booking Was Placed Successfully. Back to Trips

© 2016 - My ASP.NET Application