SOURCE CODE

```
namespace AddMarkApiMVC.Models
{
    public class Student
    {
        [Key]
        public int StudentId { get; set; }

        [Required(ErrorMessage = "First Name is required")]
        public string FirstName { get; set; }

        [Required(ErrorMessage = "Last Name is required")]
        public string LastName { get; set; }

        [Required(ErrorMessage = "Class is required")]
        public string Class { get; set; }

        [Required(ErrorMessage = "Date of Birth is required")]
        [DataType(DataType.Date)]
        public DateTime Dob { get; set; }
    }

}



public class Subject
 {
     [Key]
     public int SubjectId { get; set; }

     [Required(ErrorMessage = "StudentId is required")]
     public int StudentId { get; set; }

     [Required(ErrorMessage = "Subject Name is required")]
     public string SubjectName { get; set; }

     [Required(ErrorMessage = "Subject Total Mark is required")]
     public int TotalMark { get; set; }

     [Required(ErrorMessage = "Marks Obtained is required")]
     public int MarkObtained { get; set; }

     // Navigation property
     [ForeignKey("StudentId")]
     public Student Student { get; set; }
 }



namespace AddMarkApiMVC.Controllers
{
    public class StudentsController : ApiController
    {
        private AddMarkApiMVCContext db = new AddMarkApiMVCContext();

        // GET: api/Students
        public IQueryable<Student> GetStudents()
        {
            return db.Students;
        }

        // GET: api/Students/5
        [ResponseType(typeof(Student))]
        public IHttpActionResult GetStudent(int id)
        {
            Student student = db.Students.Find(id);
            if (student == null)
            {
                return NotFound();
            }

            return Ok(student);
        }

        // PUT: api/Students/5
        [ResponseType(typeof(void))]
        public IHttpActionResult PutStudent(int id, Student student)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            if (id != student.StudentId)
            {
                return BadRequest();
            }

            db.Entry(student).State = EntityState.Modified;

            try
            {
                db.SaveChanges();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!StudentExists(id))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }

            return StatusCode(HttpStatusCode.NoContent);
        }

        // POST: api/Students
        [ResponseType(typeof(Student))]
        public IHttpActionResult PostStudent(Student student)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            db.Students.Add(student);
            db.SaveChanges();

            return CreatedAtRoute("DefaultApi", new { id = student.StudentId },
student);
        }
        // DELETE: api/Students/5
        [ResponseType(typeof(Student))]
        public IHttpActionResult DeleteStudent(int id)
        {
            Student student = db.Students.Find(id);
            if (student == null)
            {
                return NotFound();
            }

            db.Students.Remove(student);
            db.SaveChanges();

            return Ok(student);
        }

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

        private bool StudentExists(int id)
        {
            return db.Students.Count(e => e.StudentId == id) > 0;
        }
    }
}




namespace AddMarkApiMVC.Controllers
{
    public class SubjectsController : ApiController
    {
        private AddMarkApiMVCContext db = new AddMarkApiMVCContext();

        // GET: api/Subjects
        public IQueryable<Subject> GetSubjects()
        {
            return db.Subjects;
        }

        // GET: api/Subjects/5
        [ResponseType(typeof(Subject))]
        public IHttpActionResult GetSubject(int id)
        {
            Subject subject = db.Subjects.Find(id);
            if (subject == null)
            {
                return NotFound();
            }

            return Ok(subject);
        }

        // PUT: api/Subjects/5
```

SOURCE CODE

```
        [ResponseType(typeof(void))]
        public IHttpActionResult PutSubject(int id, Subject subject)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            if (id != subject.SubjectId)
            {
                return BadRequest();
            }

            db.Entry(subject).State = EntityState.Modified;

            try
            {
                db.SaveChanges();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!SubjectExists(id))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }

            return StatusCode(HttpStatusCode.NoContent);
        }

        // POST: api/Subjects
        [ResponseType(typeof(Subject))]
        public IHttpActionResult PostSubject(Subject subject)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            db.Subjects.Add(subject);
            db.SaveChanges();

            return CreatedAtRoute("DefaultApi", new { id = subject.SubjectId },
subject);
        }

        // DELETE: api/Subjects/5
        [ResponseType(typeof(Subject))]
        public IHttpActionResult DeleteSubject(int id)
        {
            Subject subject = db.Subjects.Find(id);
            if (subject == null)
            {
                return NotFound();
            }

            db.Subjects.Remove(subject);
            db.SaveChanges();
            return Ok(subject);
        }

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

        private bool SubjectExists(int id)
        {
            return db.Subjects.Count(e => e.SubjectId == id) > 0;
        }
    }
}
```

```
namespace AddMarkApiMVC.Data
{
    public class AddMarkApiMVCContext : DbContext
    {
        // You can add custom code to this file. Changes will not be overwritten.
        // 
        // If you want Entity Framework to drop and regenerate your database
        // automatically whenever you change your model schema, please use data
migrations.
        // For more information refer to the documentation:
        // http://msdn.microsoft.com/en-us/data/jj591621.aspx

        public AddMarkApiMVCContext( ) : base("name=AddMarkApiMVCContext")
        {
        }

        public System.Data.Entity.DbSet<AddMarkApiMVC.Models.Student> Students {
get; set; }
        public System.Data.Entity.DbSet<AddMarkApiMVC.Models.Subject> Subjects {
get; set; }
    }
}
```

In web.config file

```
<connectionStrings>
   <add name="AddMarkApiMVCContext" connectionString="Data Source=BYOD-
644023\sqlexpress02; Initial Catalog=StudenMarksAPI; Integrated Security=True;
MultipleActiveResultSets=True;" providerName="System.Data.SqlClient" />
 </connectionStrings>
```