Student Survey

Made by Binary Beasts

Agenda

Team presentation

Alexandru Ștefan-Albert

Bujor Alexandru-Dumitru

Olaru Maria-Andreea

Păun Marius

Project presentation

Scope - implement an application that will facilitate surveys or consent requests to be created and filled in.

Features applicable:

- Web service with onion architecture;

- A modern layout, easy to follow;

- Landing screen where to add login name and password/Forgot password;

- Each form to have the following options: Add, Edit, Delete

- Two levels of authorization:

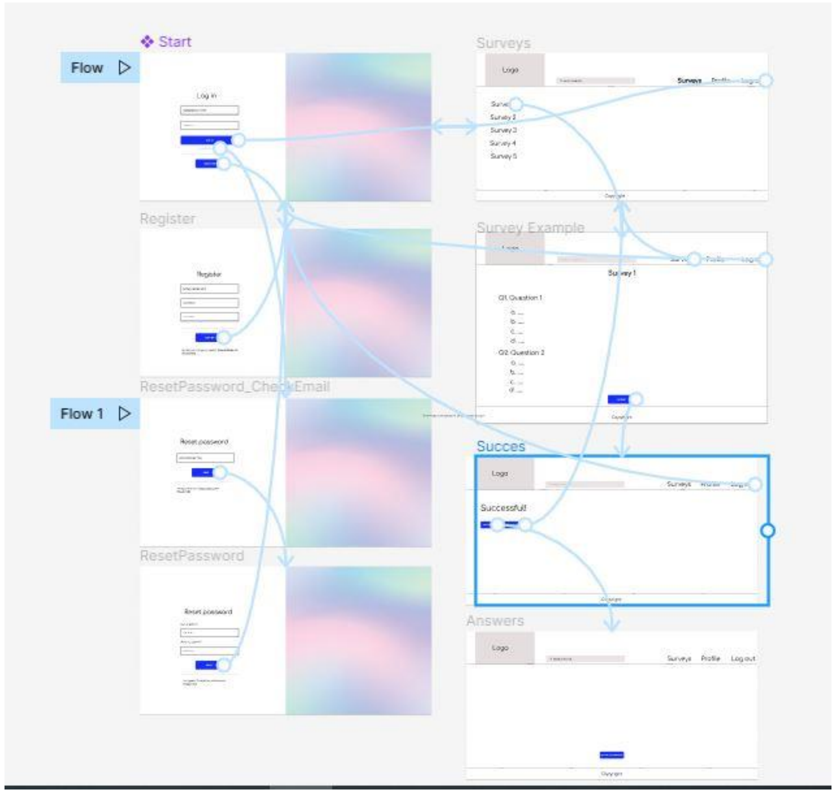
  - Administrator
  - Student

Project presentation

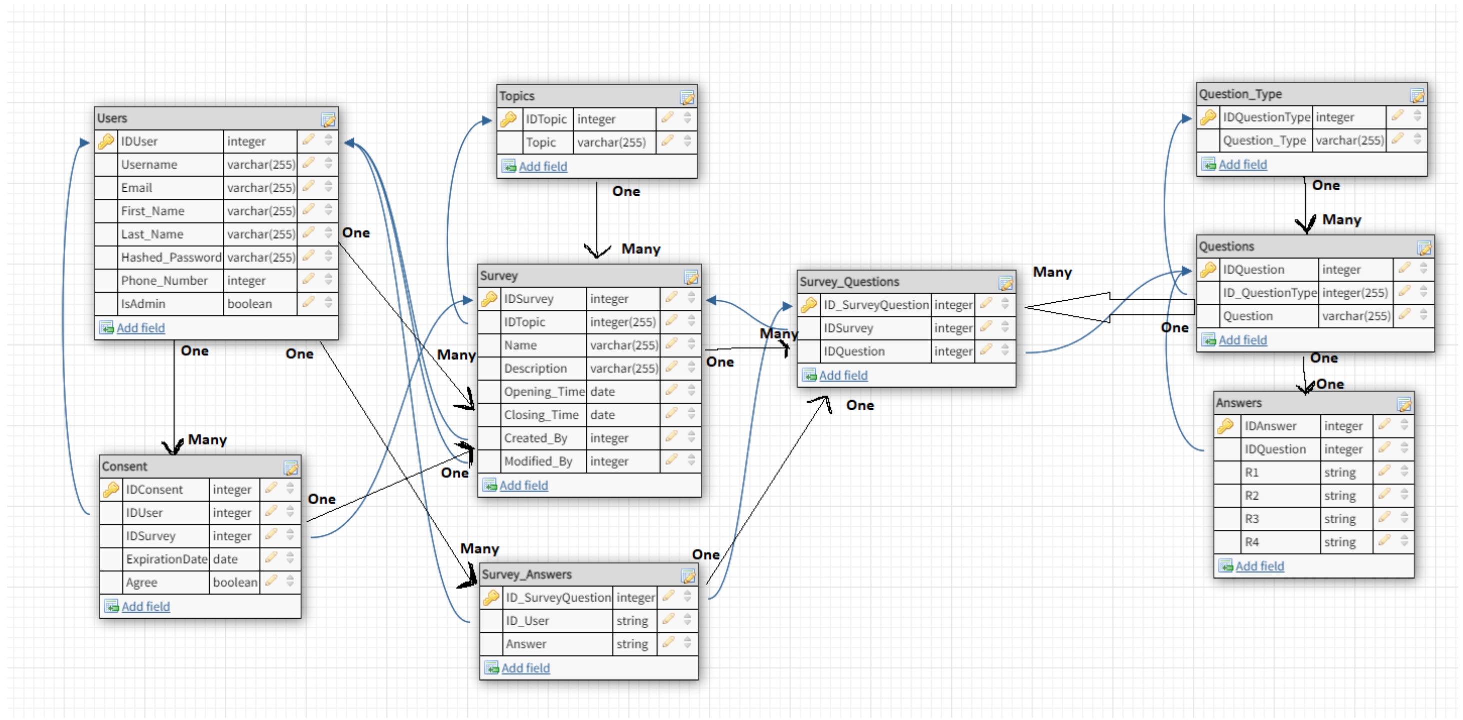
Technologies and tools used

- For **backend** - SQL Server Management Studio (SSMS), Visual Studio Community 2019, DotNet 5.0 SDK, .NET Framework 5.0, Swagger- Swashbuckle.AspNetCore 5.6.3
- For **frontend -** Visual Studio Code, VS Code extensions: Mandatory: ESLint, Prettier, Angular CLI, NodeJS
- For **testing** - Visual Studio Community 2019, ReSharper, Resharper DotCover, Selenium webdriver.

Figma Model


Start
Flow
Flow 1
Log in
Register
ResetPassword_CheckEmail
Reset password
ResetPassword
Surveys
Logo
Surveys
Survey 2
Survey 3
Survey 4
Survey 5
Survey Example
Survey 1
Q1. Question 1
Q2. Question 2
Succes
Successfull
Answers
Profile
Log out

Database layer



Users

IDUser	integer	
Username	varchar(255)	
Email	varchar(255)	
First_Name	varchar(255)	
Last_Name	varchar(255)	
Hashed_Password	varchar(255)	
Phone_Number	integer	
IsAdmin	boolean	

Topics

IDTopic	integer
Topic	varchar(255)

Survey

IDSurvey	integer
IDTopic	integer(255)
Name	varchar(255)
Description	varchar(255)
Opening_Time	date
Closing_Time	date
Created_By	integer
Modified_By	integer

Survey_Questions

ID_SurveyQuestion	integer
IDSurvey	integer
IDQuestion	integer

Question_Type

IDQuestionType	integer
Question_Type	varchar(255)

Questions

IDQuestion	integer
ID_QuestionType	integer(255)
Question	varchar(255)

Answers

IDAnswer	integer
IDQuestion	integer
R1	string
R2	string
R3	string
R4	string

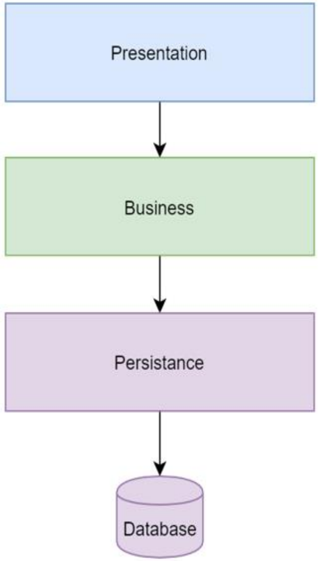
Consent

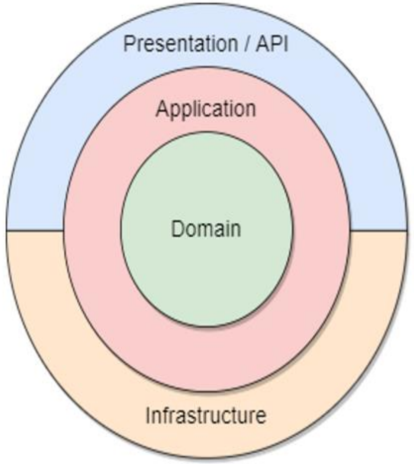
IDConsent	integer
IDUser	integer
IDSurvey	integer
ExpirationDate	date
Agree	boolean

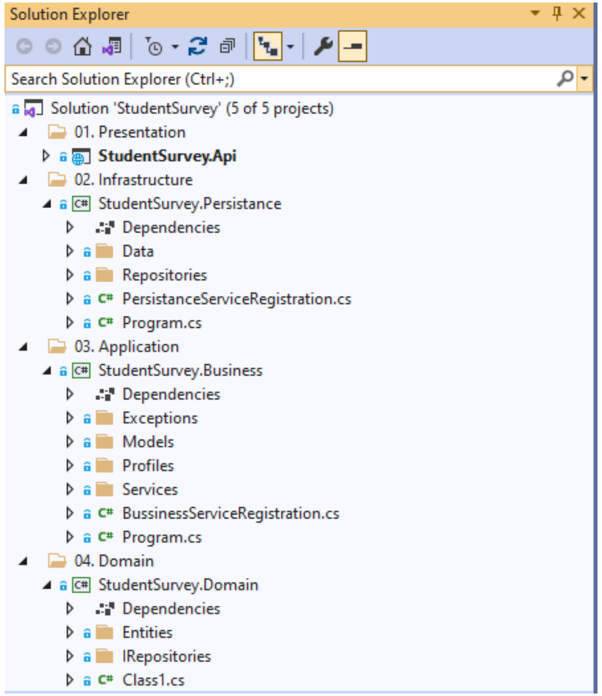
Survey_Answers

ID_SurveyQuestion	integer
ID_User	string
Answer	string

Backend code





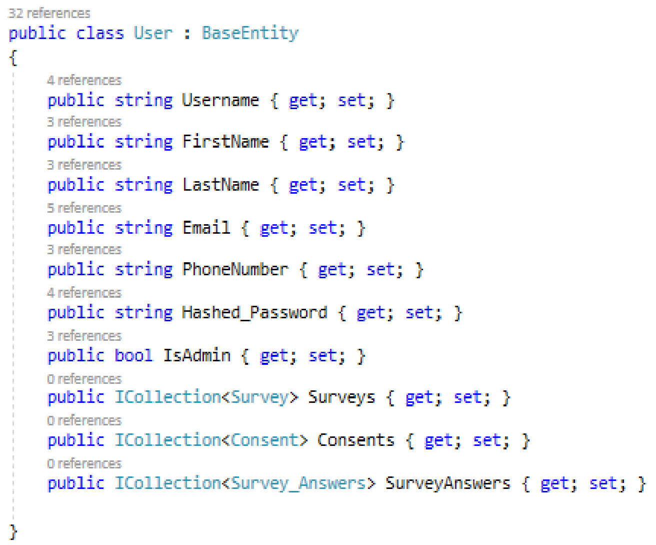

Solution Explorer

Search Solution Explorer (Ctrl+;)

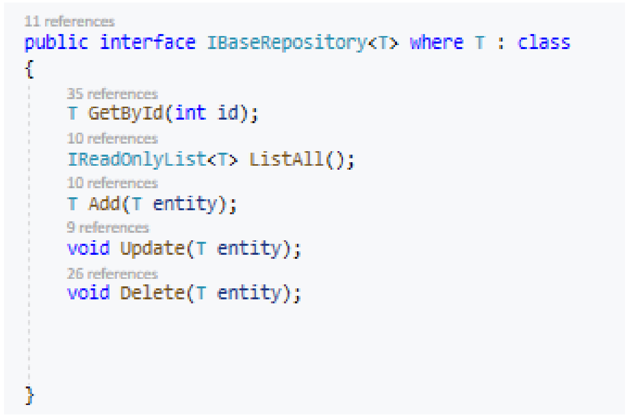
```
Solution 'StudentSurvey' (5 of 5 projects)
  01. Presentation
    StudentSurvey.Api
  02. Infrastructure
    StudentSurvey.Persistance
      Dependencies
      Data
      Repositories
      PersistanceServiceRegistration.cs
      Program.cs
  03. Application
    StudentSurvey.Business
      Dependencies
      Exceptions
      Models
      Profiles
      Services
      BussinessServiceRegistration.cs
      Program.cs
  04. Domain
    StudentSurvey.Domain
      Dependencies
      Entities
      IRepositories
      Class1.cs
```

Backend code

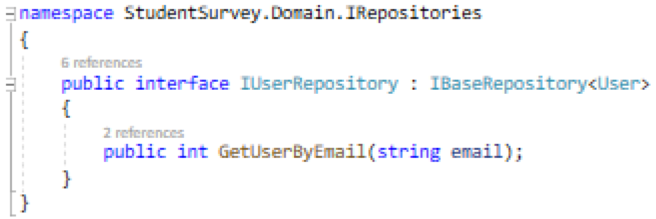
- **Domain layer** - contains the entities of the business



```
32 references
public class User : BaseEntity
{
    4 references
    public string Username { get; set; }
    3 references
    public string FirstName { get; set; }
    3 references
    public string LastName { get; set; }
    5 references
    public string Email { get; set; }
    3 references
    public string PhoneNumber { get; set; }
    4 references
    public string Hashed_Password { get; set; }
    3 references
    public bool IsAdmin { get; set; }
    0 references
    public ICollection<Survey> Surveys { get; set; }
    0 references
    public ICollection<Consent> Consents { get; set; }
    0 references
    public ICollection<Survey_Answers> SurveyAnswers { get; set; }

}
```



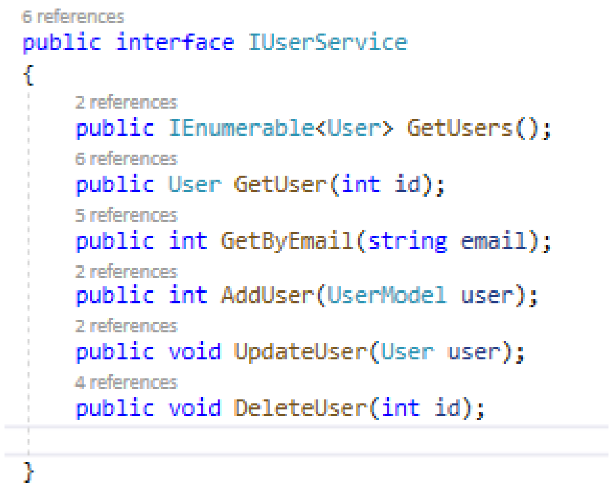
```
11 references
public interface IBaseRepository<T> where T : class
{
    35 references
    T GetById(int id);
    10 references
    IReadOnlyList<T> ListAll();
    10 references
    T Add(T entity);
    9 references
    void Update(T entity);
    26 references
    void Delete(T entity);


}
```



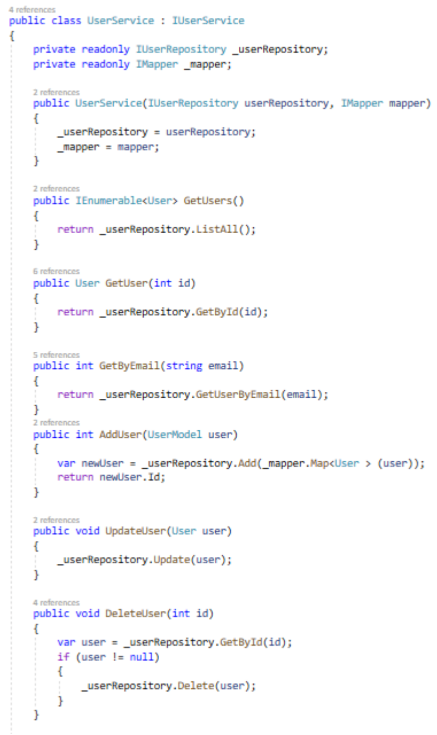
```
namespace StudentSurvey.Domain.IRepositories
{
    6 references
    public interface IUserRepository : IBaseRepository<User>
    {
        2 references
        public int GetUserByEmail(string email);
    }
}
```

Backend code

- **Application layer** - contains business logic and business rules that should strictly reflect what the real world wants from the application



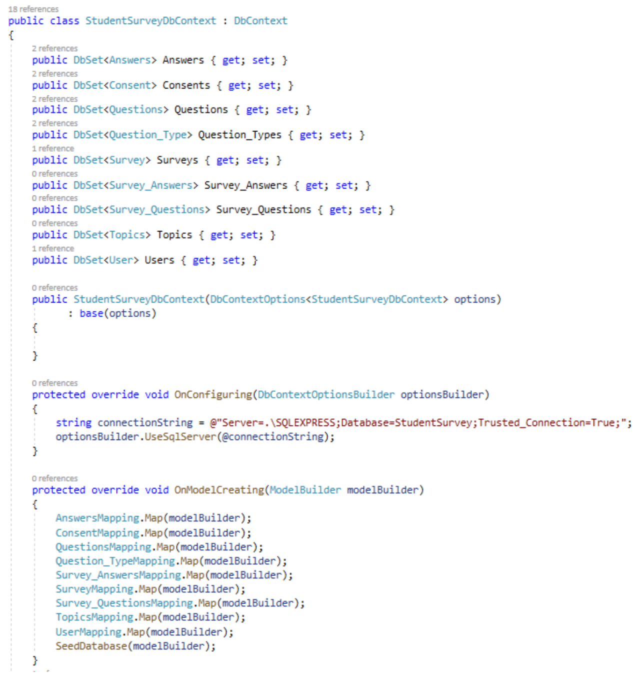
```
6 references
public interface IUserService
{
    2 references
    public IEnumerable<User> GetUsers();
    6 references
    public User GetUser(int id);
    5 references
    public int GetByEmail(string email);
    2 references
    public int AddUser(UserModel user);
    2 references
    public void UpdateUser(User user);
    4 references
    public void DeleteUser(int id);

}
```



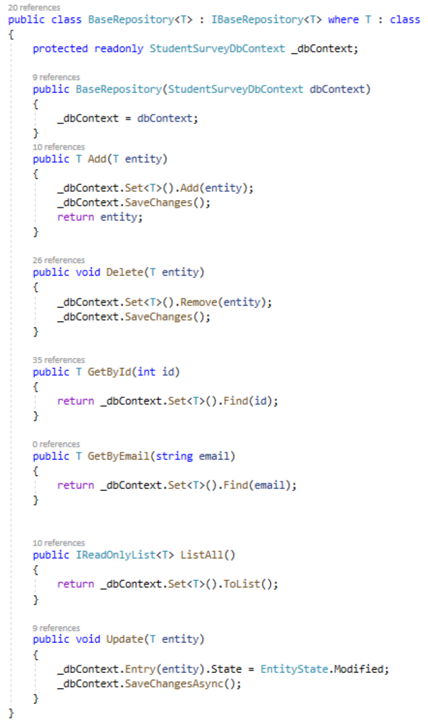
```
4 references
public class UserService : IUserService
{
    private readonly IUserRepository _userRepository;
    private readonly IMapper _mapper;

    2 references
    public UserService(IUserRepository userRepository, IMapper mapper)
    {
        _userRepository = userRepository;
        _mapper = mapper;
    }

    2 references
    public IEnumerable<User> GetUsers()
    {
        return _userRepository.ListAll();
    }

    6 references
    public User GetUser(int id)
    {
        return _userRepository.GetById(id);
    }

    5 references
    public int GetByEmail(string email)
    {
        return _userRepository.GetUserByEmail(email);
    }
    2 references
    public int AddUser(UserModel user)
    {
        var newUser = _userRepository.Add(_mapper.Map<User > (user));
        return newUser.Id;
    }

    2 references
    public void UpdateUser(User user)
    {
        _userRepository.Update(user);
    }

    4 references
    public void DeleteUser(int id)
    {
        var user = _userRepository.GetById(id);
        if (user != null)
        {
            _userRepository.Delete(user);
        }
    }
```

Backend code

- **Infrastructure layer** - this layer encapsulates any external system we depend on such as: database, file system.



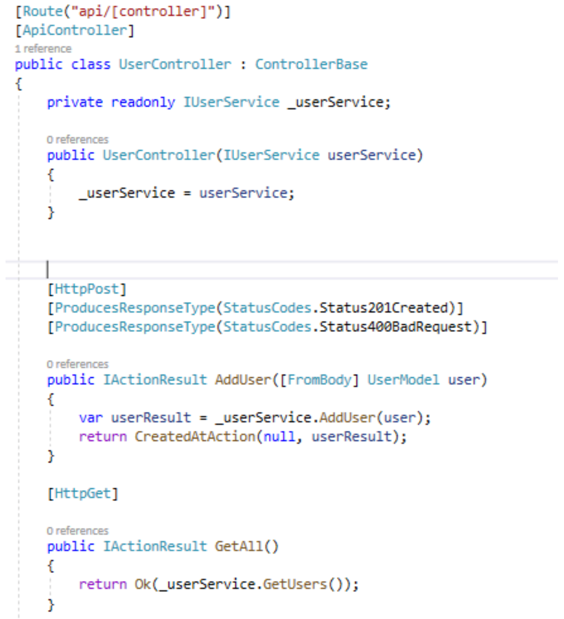
```
18 references
public class StudentSurveyDbContext : DbContext
{
    2 references
    public DbSet<Answers> Answers { get; set; }
    2 references
    public DbSet<Consent> Consents { get; set; }
    2 references
    public DbSet<Questions> Questions { get; set; }
    2 references
    public DbSet<Question_Type> Question_Types { get; set; }
    1 reference
    public DbSet<Survey> Surveys { get; set; }
    0 references
    public DbSet<Survey_Answers> Survey_Answers { get; set; }
    0 references
    public DbSet<Survey_Questions> Survey_Questions { get; set; }
    0 references
    public DbSet<Topics> Topics { get; set; }
    1 reference
    public DbSet<User> Users { get; set; }

    0 references
    public StudentSurveyDbContext(DbContextOptions<StudentSurveyDbContext> options)
          : base(options)
    {

    }

    0 references
    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        string connectionString = @"Server=.\SQLEXPRESS;Database=StudentSurvey;Trusted_Connection=True;";
        optionsBuilder.UseSqlServer(@connectionString);
    }

    0 references
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        AnswersMapping.Map(modelBuilder);
        ConsentMapping.Map(modelBuilder);
        QuestionsMapping.Map(modelBuilder);
        Question_TypeMapping.Map(modelBuilder);
        Survey_AnswersMapping.Map(modelBuilder);
        SurveyMapping.Map(modelBuilder);
        Survey_QuestionsMapping.Map(modelBuilder);
        TopicsMapping.Map(modelBuilder);
        UserMapping.Map(modelBuilder);
        SeedDatabase(modelBuilder);
    }
```



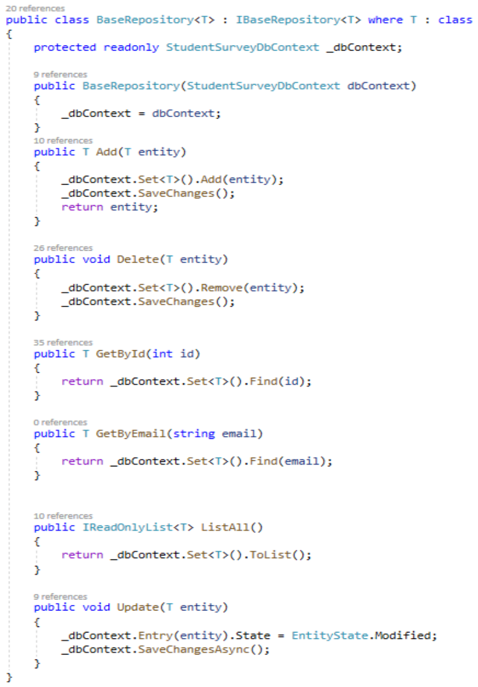
```
20 references
public class BaseRepository<T> : IBaseRepository<T> where T : class
{
    protected readonly StudentSurveyDbContext _dbContext;

    9 references
    public BaseRepository(StudentSurveyDbContext dbContext)
    {
        _dbContext = dbContext;
    }
    10 references
    public T Add(T entity)
    {
        _dbContext.Set<T>().Add(entity);
        _dbContext.SaveChanges();
        return entity;
    }

    26 references
    public void Delete(T entity)
    {
        _dbContext.Set<T>().Remove(entity);
        _dbContext.SaveChanges();
    }

    35 references
    public T GetById(int id)
    {
        return _dbContext.Set<T>().Find(id);
    }

    0 references
    public T GetByEmail(string email)
    {
        return _dbContext.Set<T>().Find(email);
    }


    10 references
    public IReadOnlyList<T> ListAll()
    {
        return _dbContext.Set<T>().ToList();
    }

    9 references
    public void Update(T entity)
    {
        _dbContext.Entry(entity).State = EntityState.Modified;
        _dbContext.SaveChangesAsync();
    }
}
```

Backend code

- **Presentation layer** -it is the first and topmost layer present in the application where users can interact with the application. In our case: swagger.

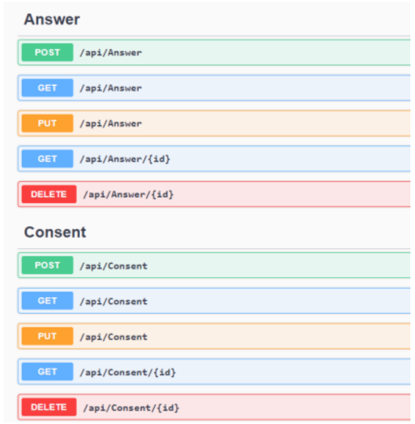


```
[Route("api/[controller]")]
[ApiController]
1 reference
public class UserController : ControllerBase
{
    private readonly IUserService _userService;

    0 references
    public UserController(IUserService userService)
    {
        _userService = userService;
    }


    |
    [HttpPost]
    [ProducesResponseType(StatusCodes.Status201Created)]
    [ProducesResponseType(StatusCodes.Status400BadRequest)]

    0 references
    public IActionResult AddUser([FromBody] UserModel user)
    {
        var userResult = _userService.AddUser(user);
        return CreatedAtAction(null, userResult);
    }

    [HttpGet]

    0 references
    public IActionResult GetAll()
    {
        return Ok(_userService.GetUsers());
    }
```



```
20 references
public class BaseRepository<T> : IBaseRepository<T> where T : class
{
    protected readonly StudentSurveyDbContext _dbContext;

    9 references
    public BaseRepository(StudentSurveyDbContext dbContext)
    {
        _dbContext = dbContext;
    }
    10 references
    public T Add(T entity)
    {
        _dbContext.Set<T>().Add(entity);
        _dbContext.SaveChanges();
        return entity;
    }

    26 references
    public void Delete(T entity)
    {
        _dbContext.Set<T>().Remove(entity);
        _dbContext.SaveChanges();
    }

    35 references
    public T GetById(int id)
    {
        return _dbContext.Set<T>().Find(id);
    }

    0 references
    public T GetByEmail(string email)
    {
        return _dbContext.Set<T>().Find(email);
    }


    10 references
    public IReadOnlyList<T> ListAll()
    {
        return _dbContext.Set<T>().ToList();
    }

    9 references
    public void Update(T entity)
    {
        _dbContext.Entry(entity).State = EntityState.Modified;
        _dbContext.SaveChangesAsync();
    }
}
```

Swagger



Answer

POST /api/Answer

GET /api/Answer

PUT /api/Answer

GET /api/Answer/{id}

DELETE /api/Answer/{id}

Consent

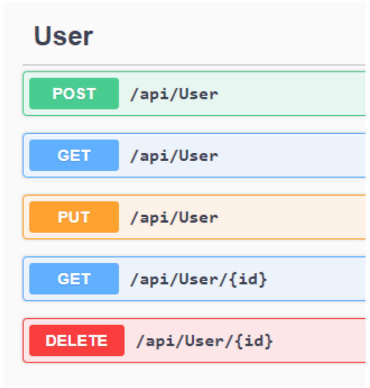
POST /api/Consent

GET /api/Consent

PUT /api/Consent

GET /api/Consent/{id}

DELETE /api/Consent/{id}



User

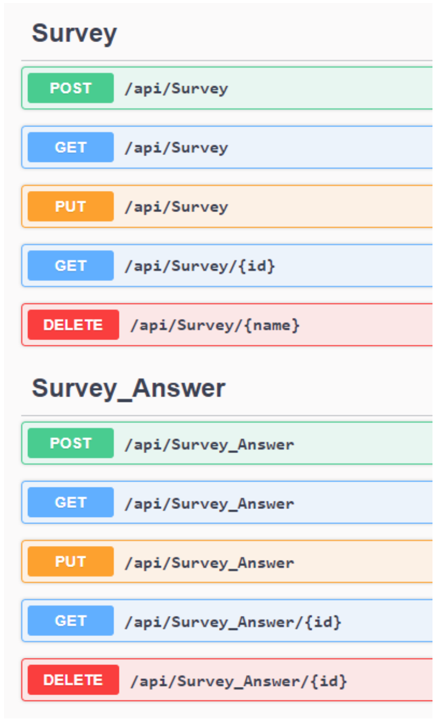
POST /api/User

GET /api/User

PUT /api/User

GET /api/User/{id}

DELETE /api/User/{id}



Survey

POST /api/Survey

GET /api/Survey

PUT /api/Survey

GET /api/Survey/{id}

DELETE /api/Survey/{name}

Survey_Answer

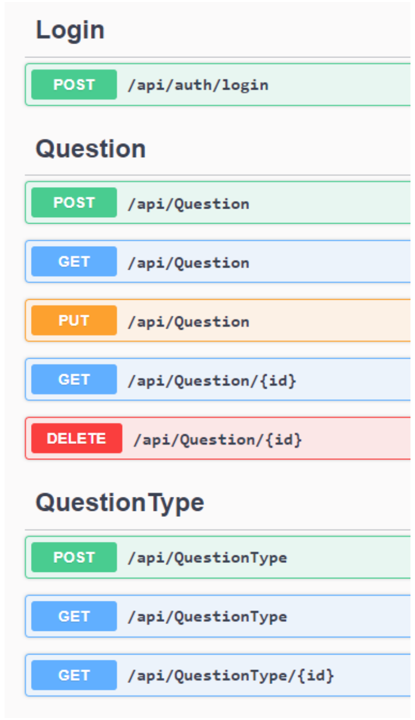
POST /api/Survey_Answer

GET /api/Survey_Answer

PUT /api/Survey_Answer

GET /api/Survey_Answer/{id}

DELETE /api/Survey_Answer/{id}



Login

POST /api/auth/login

Question

POST /api/Question

GET /api/Question

PUT /api/Question

GET /api/Question/{id}

DELETE /api/Question/{id}

QuestionType

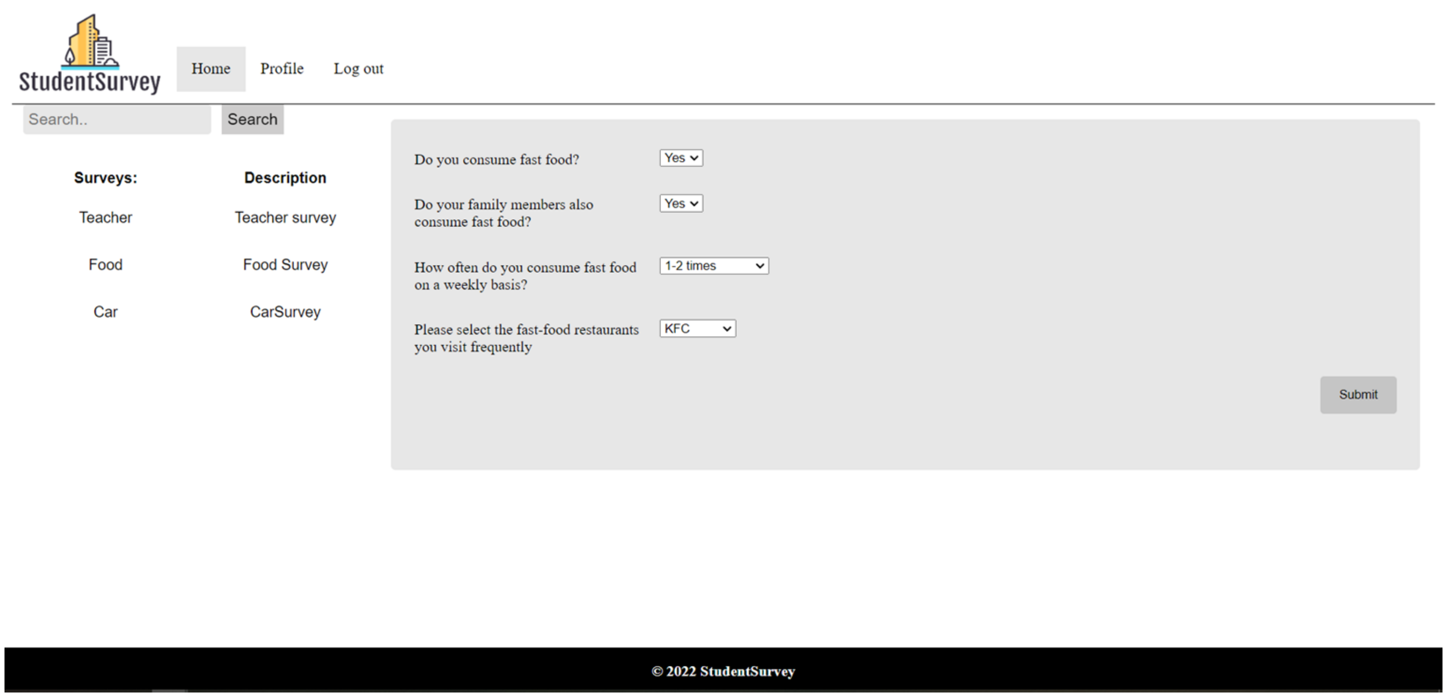
POST /api/QuestionType

GET /api/QuestionType

GET /api/QuestionType/{id}

Frontend

Interface for user



Home Profile Log out

Search.. Search

Surveys:	Description
Teacher	Teacher survey
Food	Food Survey
Car	CarSurvey

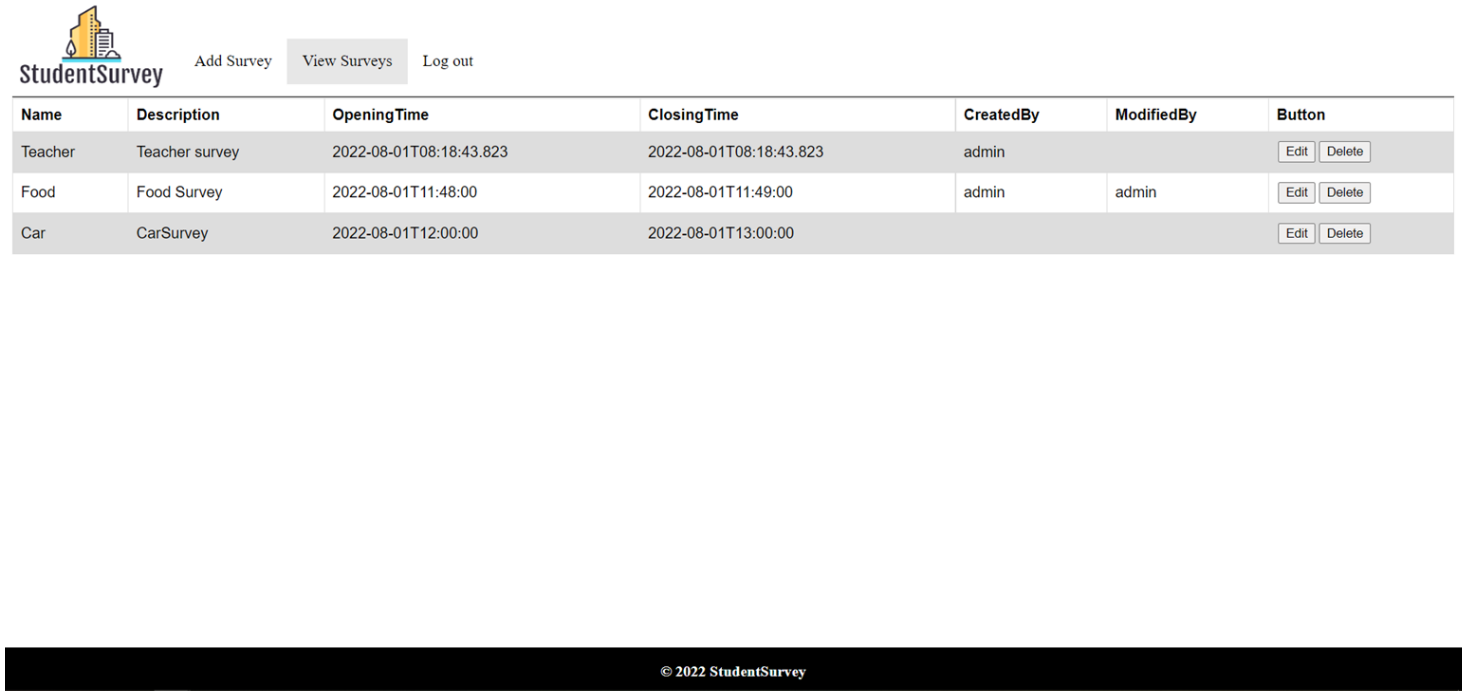
Do you consume fast food?	Yes
Do your family members also consume fast food?	Yes
How often do you consume fast food on a weekly basis?	1-2 times
Please select the fast-food restaurants you visit frequently	KFC

Submit

© 2022 StudentSurvey

Frontend

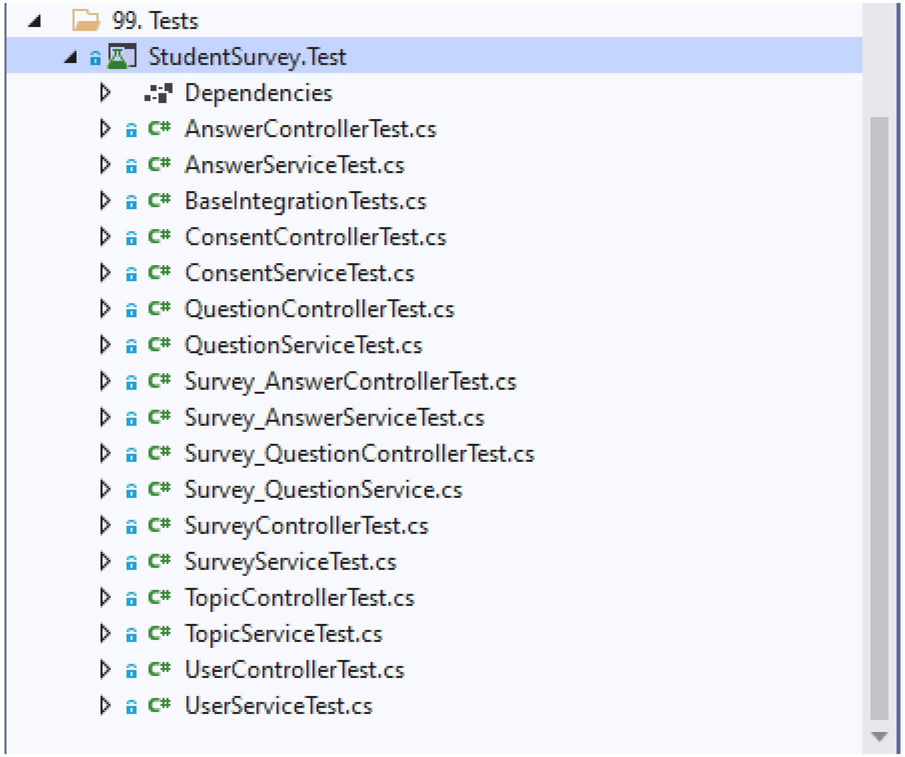
Interface for admin



Name	Description	OpeningTime	ClosingTime	CreatedBy	ModifiedBy	Button
Teacher	Teacher survey	2022-08-01T08:18:43.823	2022-08-01T08:18:43.823	admin		Edit Delete
Food	Food Survey	2022-08-01T11:48:00	2022-08-01T11:49:00	admin	admin	Edit Delete
Car	CarSurvey	2022-08-01T12:00:00	2022-08-01T13:00:00			Edit Delete

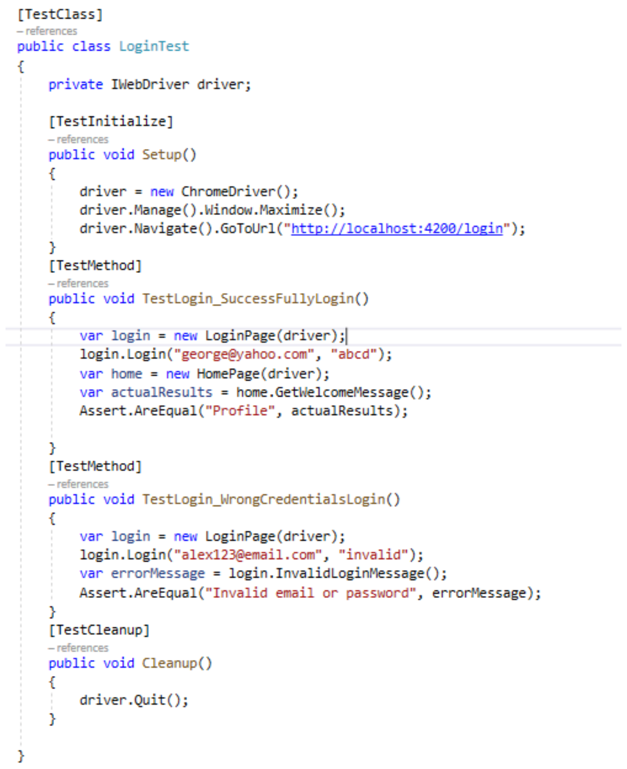
© 2022 StudentSurvey

Developer Testing



```
99. Tests
  StudentSurvey.Test
    Dependencies
    AnswerControllerTest.cs
    AnswerServiceTest.cs
    BaseIntegrationTests.cs
    ConsentControllerTest.cs
    ConsentServiceTest.cs
    QuestionControllerTest.cs
    QuestionServiceTest.cs
    Survey_AnswerControllerTest.cs
    Survey_AnswerServiceTest.cs
    Survey_QuestionControllerTest.cs
    Survey_QuestionService.cs
    SurveyControllerTest.cs
    SurveyServiceTest.cs
    TopicControllerTest.cs
    TopicServiceTest.cs
    UserControllerTest.cs
    UserServiceTest.cs
```

Automating Testing



```
[TestClass]
- references
public class LoginTest
{
    private IWebDriver driver;

    [TestInitialize]
    - references
    public void Setup()
    {
        driver = new ChromeDriver();
        driver.Manage().Window.Maximize();
        driver.Navigate().GoToUrl("http://localhost:4200/login");
    }
    [TestMethod]
    - references
    public void TestLogin_SuccessFullyLogin()
    {
        var login = new LoginPage(driver);
        login.Login("george@yahoo.com", "abcd");
        var home = new HomePage(driver);
        var actualResults = home.GetWelcomeMessage();
        Assert.AreEqual("Profile", actualResults);

    }
    [TestMethod]
    - references
    public void TestLogin_WrongCredentialsLogin()
    {
        var login = new LoginPage(driver);
        login.Login("alex123@email.com", "invalid");
        var errorMessage = login.InvalidLoginMessage();
        Assert.AreEqual("Invalid email or password", errorMessage);
    }
    [TestCleanup]
    - references
    public void Cleanup()
    {
        driver.Quit();
    }

}
```

Project DEMO