Project

Item	Description
Goal	Getting a practical experience in • Development of layered large application by steps (every module) • Using C#/.NET programming features in software development
Task	• Develop componential layered application for airport management. • provide: • **view** of the airline flight information about arrivals and departures (separately). It should reflect the information about the arrival (departure) date and time, flight number, city/port of arrival (departure), terminal, flight status (check-in, gate closed, arrived, departed at, unknown, canceled, expected at, delayed, in flight), gate • **view** of the flights pricelist with the class prices • **view** of the passengers list. It should reflect the information about the flight number, passenger first name, second name, nationality, passport, date of birthday, sex, class (business, economy). Only for company staff • **insert, delete and update** of this information. Only for company staff • **search** by the flight number, price, first and second name, passport, arrival (departure) port of and information output in the specified format. Clients information is only for company staff
Preparation	Students use labs and lectures source code of each module
Execution\ methodology	Remind student should apply his or her knowledge in: • Topics of project module • Show students and discuss general architecture of application Students have to execute following steps: • Step 1 – Module 1(Частина 1) Business logic implementation. Separate business logic and user interface (they must be in different methods: business-logic method should not contain any I/O operation) • Declare structs for flight information, flights pricelist and passage list views • Represent data as arrays (simple types and structures) • Business-logic methods: view arrivals, view departures, view flight pricelist, view passengers list, search flight number, search first and second name. Methods should take parameters (data for manipulations and return results; for example, search flight number must return flight with input number) • Implement interface methods for every business-logic method

Item	Description
	• All functionality could be implemented in Program.cs; but in different methods • Develop simple menu as different interface methods • Develop simple parts of functionality using arrays, casting and type conversions, loops, switch, read/write from/to console, string format, Console class properties • Step 2 – Module 2 (Частина 1) Business logic and data layer implementation. Separate UI, BL and DL (they must be in different classes; create folders for UI, BL and DL). Different layers should interact between each other as in classic layered architecture. • Declare classes for data (change code with structs and arrays) as in EF approach • Declare clases for methods functionality (developed in module 1 Частина 1) and move them to new classes. Change code if it is necessary • Declare method signatures for all not implemented functionality (searching and CRUD operations). Discuss with students parameters • Declare and implement separated class with methods for menu • • Declare classes for parts of functionality using classes (with methods and properties), interfaces, arrays, loops, switch, read/write from/to console • Step 3 – Module 4 (Частини 1) • Use collections/ generics instead of arrays. Change the code • Implement other searching methods • Implement CRUD operations • Change code if it is necessary • Use classes (with methods and properties), interfaces, collections and generics instead of arrays, observer pattern and events to get info about current arrival, departure of planes, loops, switch, read/write from/to console • Implement IComplarable, IComparer, IEnumerable in searching • Step 4 – Module 5 (Частина 1) • Create new project DAL (as dll) • Design database for application • Declare classes in DAL as EF Code First approach and write code for DB connection • Declare classes for default data (use migrations) • Connect BL project and DAL: data from EF classes will be copied to DL classes of main project • Change code if it is necessary • Use LocalDB SQL Server (or other type of Server), Entity Framework, CRUD with LINQ using, read/write from/to console • Step 5 – Module 4 (Частина 2) • Design and develop Windows desktop User Interface (WPF) as new WPF UI project

Item	Description
	• Design application layers as MVVM • Create new project BAL (as dll) and move there BL and DL source code • Use developed DAL project • Connect UI, BAL and DAL • Use XAML, LocalDB SQL Server, Entity Framework • Step 6 – Module 5 (Частина 2) • Design and develop Web User Interface (ASP.NET MVC) as new MVC UI project • Connect UI and BAL • Use Model-View-Controller design, javascript/ iQuery/ ajax, LocalDB SQL Server, Entity Framework
Evaluation	• the program must work correctly; • pay attention to the debugging • pay attention to the useful tips • Step 1 – Module 1(Частина 1) • Declared structs for flight information, flights pricelist and passage list views • Implemented business-logic methods: view arrivals, view departures, view flight pricelist, view passengers list, search flight number, search first and second name. Methods take parameters (data for manipulations and return results) • Implemented interface methods for every business-logic method • Separated business logic and user interface parts in different methods (business-logic method should not contain any I/O operation) • Developed simple menu as different interface methods • Used arrays, casting and type conversions, loops, switch, read/write from/to console, string format, Console class properties • Step 2 – Module 2 (Частина 1) • Declared classes for data (change code with structs and arrays) as in EF approach • Declared classes for methods functionality (developed in module 1.1) and moved to new classes • Declared method signatures for all not implemented functionality (searching and CRUD operations) • Declared and implemented separated class with methods for menu • Created folders for UI, BL and DL and classes moved to these folders • Different layers interacted between each other as in classic layered architecture • Used classes, interfaces, arrays, loops, switch, read/write from/to console • Step 3 – Module 4 (Частини 1) • Used collections/ generics instead of arrays

Item	Description
	• Implemented other searching methods • Implemented CRUD operations • Used collections and generics, observer pattern or events to get info about current arrival, departure of planes, loops, switch, read/write from/to console • Implemented IComplarable, IComparer, IEnumerable in searching (possible to use some of them) • Step 4 – Module 5 (Частина 1) • Created new project DAL (as dll) • Designed database for application • Declared classes in DAL as EF Code First approach and write code for DB connection • Declared classes for default data (used migrations) • Connected BL project and DAL • Used LocalDB SQL Server (or other type of Server), Entity Framework, CRUD with LINQ using, read/write from/to console • Step 5 – Module 4 (Частина 2) • Designed and developed Windows desktop User Interface (WPF) as new WPF UI project • Designed application layers as MVVM • Created new project BAL (as dll) and moved there BL and DL source code • Connected UI, BAL and DAL • Used XAML, LocalDB SQL Server, Entity Framework • Step 6 – Module 5 (Частина 2) • Designed and developed Web User Interface (ASP.NET MVC) as new MVC UI project • Connected UI and BAL • Used Model-View-Controller design, javascript/ iQuery/ ajax, LocalDB SQL Server, Entity Framework
Closure	Issues discussion, on demand
Reporting	No special reporting