Coursework. Speech

Number of slide. Speech text, word [phonetic transcription].

1. The topic ['tɔpɪk] of my coursework is visual graph [ɡræf] editing ['edɪt]. So I had to develop an application which is in fact a visual graph editor.
2. There’s the short list of requirements for the application on this slide.  
   The application should have the very basic features. It should allow user to add nodes, remove nodes, move nodes, specify ['spesɪfaɪ] captions for the nodes, draw links between nodes and so on. The possibility of storing documents ['dɔkjumənt] in files is also an important requirement.
3. The application is written in C# [ʃɑːp] programming language. So it’s based on the Microsoft .NET framework. I used Microsoft Visual Studio ['stjuːdɪəu] Express [ɪk'spres] 2012 integrated development environment. Since my application is a visual editor, the user interface is an important part of it. To build the user interface of my application I used Windows Presentation Foundation framework.
4. I also used git [ɡít] source code management system with a graphical frontend. I stored my git repositories [rɪ'pɔzɪt(ə)rɪ] online with GitHub hosting [həust] service. And I used NLog logging library to debug my application.
5. Here’s how the application window looks. You see some document being displayed. It contains several nodes and links. All operations are accessible [ək'sesəbl] through the main [meɪn] menu ['menjuː] and the context ['kɔntekst] menus of the nodes.
6. This application provides the very basic functionality. The user can add new nodes to the document, then remove any node. Nodes can be dragged across the visual field. It is possible to rename nodes; draw links between nodes, then again remove those links. Large documents can be scrolled using the scroll bars.
7. One of the important features of the application is the ability to store graphs inside XML [eks em ˈel] documents so they can be easily accessed, for example [ɪg'zɑːmpl], from other applications. These documents have the following structure ['strʌkʧə]: in the root element named “content” there are NodeControl [kən'trəul] and LinkControl sub-elements. Each of these xml elements represents a node or a link respectively. NodeControl elements have the following attributes ['ætrɪbjuːt]: text, position stored as left and top offsets from the left top corner of the document area, and a unique identifier, while each LinkControl element has two attributes which store unique IDs of the nodes which are interconnected by this particular link.
8. While working on this project I got to use the Windows Presentation Foundation framework. So in my presentation I wanted to mention some of its features. It supports hardware ['hɑːdweə] acceleration [əkˌselə'reɪʃ(ə)n] and it is resolution independent [ˌɪndɪ'pendənt].
9. It also supports styles and it’s actually possible to create interesting-looking user interfaces with it. Like this one. Even though the application I developed is not of great complexity [kəm'pleks, kəm'pleksətɪ], this framework saved me a lot of time.
10. One of the key features of the Windows Presentation Foundation is that it provides means for separating user interface design from the application source code responsible for application logic. The user interface should be specified in XAML, an xml-like markup language. Here’s an example of it from my application. This code specifies a part of the main window.
11. Windows Presentation Foundation introduces dependency [dɪ'pendən(t)sɪ] properties and provides means for connecting properties of different objects like, for example, some data and a window control which displays this data, so when the data changes, the visual representation of this data gets updated immediately. This works both ways. When the user changes the data using the visual control, this data actually gets changed and updated in all other places where necessary in case they were connected properly to the dependency property of this control. This concept is called Data Binding ['baɪndɪŋ]. Dependency Properties and Data Bindings can be used in many different ways, they are not limited to simple cases. It’s actually a complex system with many additional features. These are pieces of source code from my coursework involving the dependency properties. Using them makes the application more complex to some point.
12. There are also two notable features of the C# language which I could make use of while working on the project. They are still relatively ['relətɪvlɪ] new while I believe they are well-known among C# developers. Microsoft LINQ (читается “link”) is a set of special features for searching lists and processing various ['veərɪəs] data. For example, this expression searches for links connected to the node. Of course, the usage of LINK is not limited by such simple cases. For example it also can map .net objects to relational database tables and do many other things related data processing.
13. And another feature of the CSharp programming language is the lambda ['læmdə] expression. Which is, in fact, a short declaration of an anonymous function which can be constructed inside another function and passed to other functions as an argument ['ɑːgjəmənt] so that they may call it later when necessary. Some other programming languages have this feature. This piece of code is also taken from my project and it sets dependency property value for every element of the list which matches specified type. The lambda expression is in the green ['rektæŋgl] rectangle. Perhaps it is possible to think of a better example of using lambda expression, but I believe it is appropriate here because it is just a shortened declaration of a function. And here I declare a small procedure. Here I’m telling the code what to do with the elements matching the specified criteria [kraɪ'tɪərɪə]. Their ID property should be assigned to a new value. Well, …
14. that’s all. This brings me to the end of my presentation. Thank you for your attention.