Zhewen Hu

Windows Phone 7 Application

Student Helper

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Thesis

14 May 2013

Helsinki Metropolia University of Applied Sciences

Author(s) Title Number of Pages Date	Zhewen Hu Windows Phone 7 Application Student Helper 34 pages 14 May 2013
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Jaana Holvikivi, Principal Lecturer

This final year project was based on the Windows Phone 7 mobile operating system with the main goal to build a Windows Phone 7 mobile application for students. A student is to be able to manage schedules, record class notes and course assignments with the help of the application. The project was carried out on Nokia Lumia 800.

Education software is becoming popular on mobile platforms. The aim of the project was to develop course management software called Student Helper. The software helps students to manage scheduling their course work and to record homework as well. Besides, students can write notes into the mobile phone. It is convenient for students because students do not need a pen or a pencil to record notes and homework. They do not need to worry about losing notes which are stored in the phone's memory. They can store information in the mobile phone, even when there is no internet connection. This project was particularly designed for offline course management.

The application was successfully designed and implemented with standard software engineering development methods. The development phases included analysis, design, implementation, and testing. The product satisfied the general user requirements. This thesis also discusses the upsides and downsides of the project.

Keywords	Windows Phone 7, C#.NET, student course work management

Contents

Abbreviations and Acronyms

C#	Microsoft's Programming Language
IOS	IPhone Operating System
JSON	JavaScript Object Notation
SDK	Software Development Kit
UI	User Interface
UML	Unified Modelling Language
WIFI	Wireless Fidelity
XAML	Extensible Application Markup Language
.NET Framework	Microsoft's Software Framework

1 Introduction

In the smart mobile phone age, the role of the phone has changed; it is not only a pure phone but a mobile computer as well, it can function as a music player, a mobile navigator or it can even be a game console. A phone can be used to access the Internet, listen to music, send emails and so forth. The Windows Phone OS is one of the leading operating systems (OS) in the mobile OS market. The Windows Phone OS is the fourth largest mobile operating system in the world market. Windows Phone 7 and 8 are new mobile operating systems and successors to the earlier Windows Mobile OS. Compared to the Android and Apple IOS OS markets, there are far less applications on the Windows Phone application market. Thus, the Windows Phone application market has a relatively bigger market growth potential.

The 21st century is the information technology era; electronic education software and online education courses help a growing number of people to acquire new knowledge. There is no doubt that the Internet helps students and teachers to enhance their efficiency. Educational software helps teachers and students to interact in a novel way, and not only in the PC realm. It is also convenient to use a mobile phone to assist the students, on one hand, to learn more, and on the other hand, to have a better course arrangement in the mobile internet realm. Student information systems are widely used in schools. As time goes by, there will be more and more software to help students to improve their study. The goal of this project was to build useful course management software for students, based on Windows Phone 7 OS. Additionally, the thesis offers an overview about Windows Phone OS. The project is a student information system called Student Helper.

Student Helper is designed to perform three main functions, which are all targeted at students. It allows users to manage their schedules, add or modify course homework, and to record or modify course notes. All the data is stored on the phone's local disk. Thus, the smart phone does not rely on the Internet; the application can function without an Internet-connection environment. At many universities, students use online schedule systems. The application is based on the Windows Phone 7 platform, so if the students have a Windows phone, they can install the application. No pen and paper is then needed to record notes, schedules and home assignments. Therefore, the use of the application can enhance the students' study efficiency, and reduce the weight of

their bags. Additionally, the students could use less paper so that the application indirectly contributes to the protection of the environment. Overall, this application can be esteemed as environmentally friendly and beneficial for students.

The application is developed with Microsoft Visual Studio 2010 and Microsoft Expression Blend 4; the programming languages are C# for the logical code part and XAML (Extensible Application Markup Language) for user interface (UI) design. To gain usability and better user experience, some Silverlight toolkits were used in the application so that it would have more functions.

2 Windows Phone Operating System

2.1 System History

The predecessor of the Windows Phone OS was Windows Mobile OS. Unfortunately, it had only a small market share in the early smart phone age. Microsoft had to launch a new mobile OS. The successor to Windows Mobile OS, Windows Phone OS, plays this role, aimed at gaining more market share in the mobile phone world.

Although the Windows Phone OS was launched later than its competitors, it is still a magnificent mobile OS. It supports all the same features that Android and IOS have: email, social networking, navigator, multi-touch, WIFI and so forth. It also offers a novelty: a totally new kind of UI code named METRO UI. The Windows Phone 7 was first released in 2010. [1] After WP7, Microsoft launched newer OSs based on the WP7 architecture, such as WP7.5 Mango OS and WP7.8. The Windows Phone 8 was released in 2012. It is the most recent Windows Phone OS version. Compared to WP7, WP8 OS has enhanced functions and updated advanced settings [2]. WP8 is a totally new generation mobile OS, because it has a new computer architecture, based on a Windows NT kernel.

2.2 User Interface

To introduce a visible difference from the other two most popular mobile platforms, Android and IOS, Microsoft designed a new style user interface with new typography, Metro. It was initially used in Windows Phone. The most prominent character of Metro UI is that it shows a lot of information to users.

There are five principles in the Metro design. The first principle is to focus on a clean, light looking, open, and fast system. The second principle in Metro design is to concentrate on content. It means that Metro should always present some content. The third principle concentrates on gestures supporting the best possible ever multi-touch user experience. The fourth principle is to focus on seamless integration between hardware and software. The last principle is that the application should be alive, soulful and emotional. [3, 12] Although the name Metro was abandoned by Microsoft because of trademark dispute, the name “Metro” is still used in this thesis.

The icon in the main user interface always shows some content information to the users. Figure 1 shows the common Metro UI main menu:

Figure 1. Metro UI [4].

The picture in Figure 1 shows that there are 4 unread messages and 25 e-mails. As can be seen, the UI design style is informative and concise, so that users can easily grasp the current status. Metro offers not only a concise UI but also motions and transitions which increase the interactivity, usability and visualization for users. Metro UI is now deployed in Windows Phone 8, Windows 8, and Microsoft's official website etc. Metro UI will most likely be Microsoft's main theme in the future.

2.3 Panorama Style

The Panorama style UI is only shown in WP. This kind of UI style cannot be found in any other mobile phone OS. It gives a user the experience of a continuous view in contrast to a paged view. The idea of the Panorama design allows a user to control the field of view by touching the phone screen to view the desired content. The Panorama can also be implemented on full screen and treated as a navigation mode in an application.

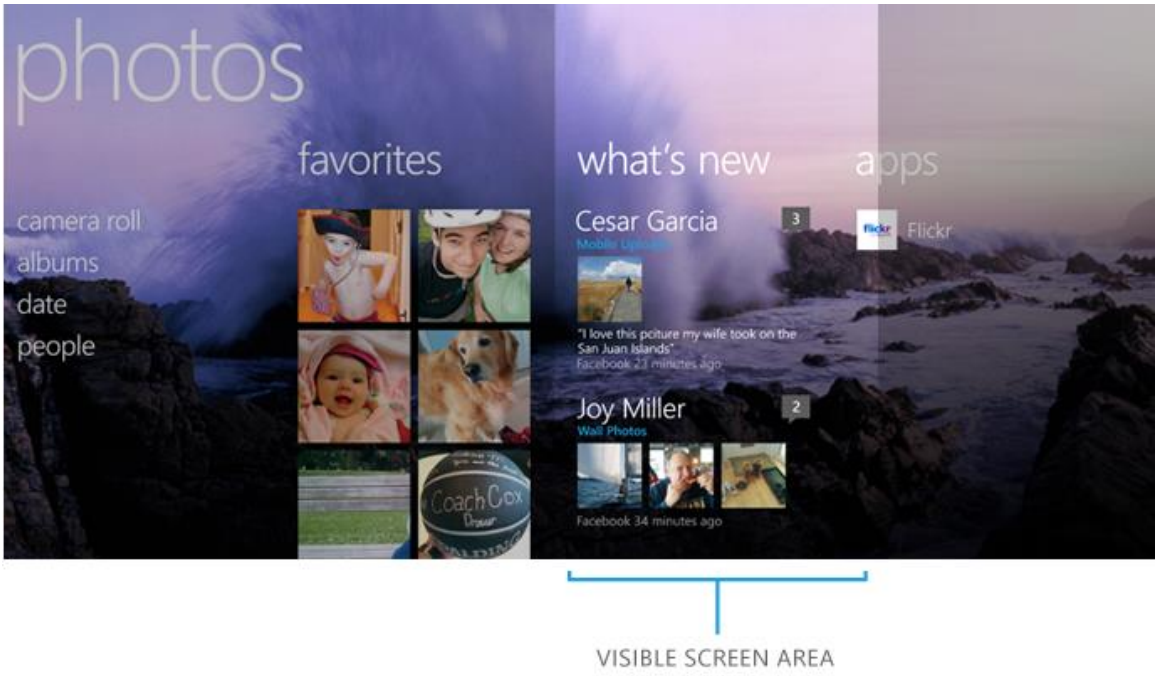

Figure 2. Panorama style [5].

Figure 2 shows the Panorama UI style. The visible screen area is a part of the panorama background, and the user can move the screen to left or right, and decide which part to read. The UI supports casual browsing back and forth, and enhances the usability and interactivity.

3 Project Design

3.1 User Requirement Analysis

In general, a software project starts with user requirement analysis. The purpose of this final year project is to build an application called Student Helper. Every student may encounter the following situation: a teacher talks about an important thing but the student does not have pen or paper. In this kind of a situation, a student may forget an important note or homework. Student Helper is designed for three main functions, all specifically intended for students. Student Helper allows the user to manage schedules, add or modify course homework, and to record and modify course notes.

Imagine a scenario where a user needs to use the application to study. As the first step, the user can add the course information into the application. Then the application stores the information in a mobile storage area, and finally, the user can read the course information from the application. The user can also record class notes and course assignments and read the notes and assignments from the phone. All the data is stored on the phone's local disk. The possible user group is all students.

The user needs a fast and efficient application. Moreover, a concise UI should be designed. For this, the user can see three big tiles on the main page of the application. They are schedule, notebook and homework parts, respectively.

3.2 UML Design

Software design is an important step in a general software development process. It depicts the overall architecture of the system and shows the function of the system. Software architecture design is focused on understanding how to organize the system and designing the whole structure of the system [6,148].

UML (Unified Modeling Language) is a standard language for object-oriented design in software engineering. This project is developed by using C# and XAML languages. Obviously, UML design was used in this project.

In general, UML is used for software modelling and architecture design. Furthermore, UML is used to design the application architecture. The project has several UML dia-

grams to describe different scenarios in a specific environment. In this project, the UML modeling tool Astah Professional was used. In the following parts, use case diagrams, and class diagrams are shown.

3.2.1 Use Case Diagram

In UML, the use case diagram represents the functionality provided by the system. It describes the basic functions of the application. In the use case diagram, the user is defined as an actor. The user can add, edit and delete a course; add, edit and delete notes; add, edit and delete homework.

The use case diagram designed is shown in Figure 3 below:

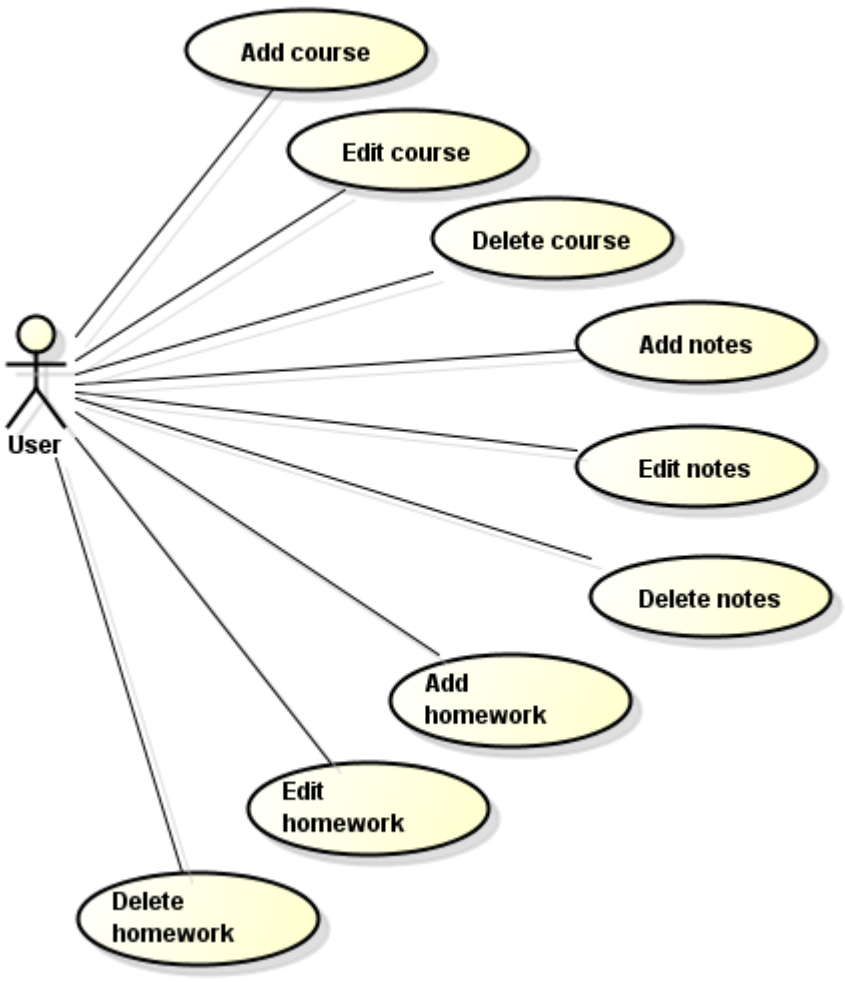

Figure 3. Use case diagram

The design for the application is the following: when the user clicks the add course button, the course information can be added, the user saves the information. The user can edit the course when he/she wants to, and, naturally, he/she can also add new courses. If there is no course stored in the schedule, the user cannot add/edit notes and homework because notes and homework is meaningless if no course exists. Therefore, the user can edit a course and add and edit the notes and homework of a course only once the course is added.

3.2.2 Class Diagram

A class diagram in UML is a structure diagram that describes the structure of the sequences of a system, showing the classes, attributes, operations and relationships among the classes of the system. In this project, the objects are course, note, and homework. Thus, these three objects must be modelled when developing the Student Helper.

The first of the three, the schedule and course description is defined in Figure 4 below:

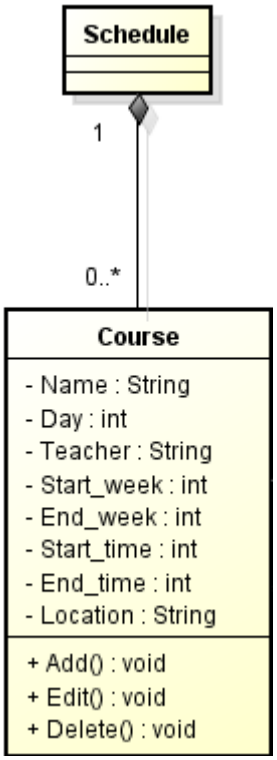

Figure 4. Class diagram: Entity Schedule and Course

The two entities are schedule and course. A schedule can contain no courses or many courses. The relationship between the two is composition. The course entity is defined by the following attributes: Name, Day, Teacher, Start_week, End_week, Start_time, End_time, and the three operations: Add, Edit and Delete.

The second entity, note, means class note. It is defined in Figure 5 below:

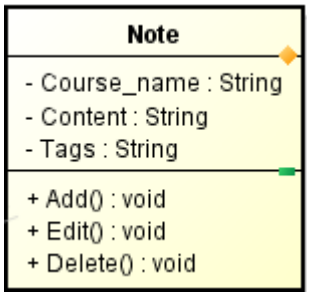

Figure 5. Class diagram: Entity Note

The entity is note. A course can contain no notes or many notes. The relationship between them is composition. The note entity is defined by three attributes: Course_name, Content, Tags, and the three operations are: Add, Edit and Delete.

The third entity is homework, described in Figure 6 below:

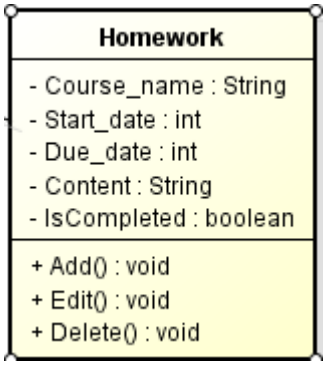

Figure 6. Class diagram: Entity Homework

The homework entity is defined by the following attributes: Course_name, Start_date, Due_date, Content and IsCompleted, and the operations are: Add, Edit and Delete.

After the creation of the entities, the full class diagram is as shown in Figure 7 below:

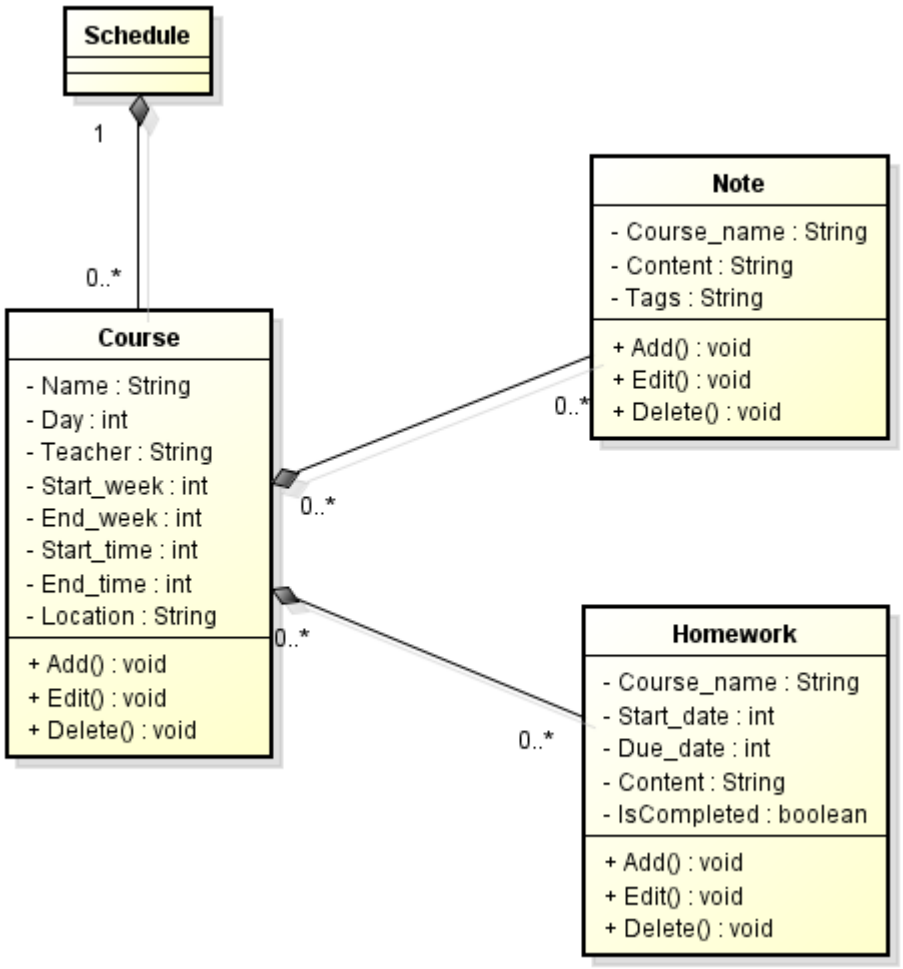

Figure 7. Overall class diagram.

The entities Note and Homework belong to the entity Course, the entity Course belongs to the entity Schedule. The relationship among the four entities is composition. The number of the courses, notes and homework can be 0 or many, whereas there can be only one schedule. Therefore, the mapping relationship between the four entities is clear and concise.

4 Project Implementation

4.1 Tools

A Windows Phone application is generally designed with Visual Studio 2010 and Microsoft Expression Blend 4. In the design part, Astah Professional was used in the project as well. Moreover, other Windows Phone toolkit controls needed to be imported in the project to gain better usability and to enhance the user experience. One of them was Coding4Fun, a free Windows Phone toolkit available to all developers.

4.2 Detailed Implementation

In the following, the details of the implementation of the four subsections of the project are shown.

4.2.1 Implementation of the Main Page

To implement the main page UI, some Windows Phone official controls were used in the application. There is a control called Coding4Fun Tile control that enables developers to add informative tiles to the application. The tiles can show images, titles, and messages, and give notifications. After adding the Coding4Fun Tile control, the application would be more informative and interactive. Figure 8 is Coding4Fun Tile control sample.

Figure 8. Coding4fun Tile Sample [6]

To use the tiles, the most important step is to add “Coding4Fun.Phone.Controls.dll” assembly to the project. For easy distinguishability, the prefix “C4F” is used as shown in Listing 1. Also, XAML code was added to the project’s main page file.

```
xmlns:C4F="clr-
namespace:Coding4Fun.Phone.Controls;assembly=Coding4Fun.Phone.Co
ntrols"
```

Listing 1. Definition of the reference

As Listing 2 illustrates, XAML code was added inside the StackPanel block.

```
<StackPanel Orientation="Horizontal" Margin="0,0,0,0"
Grid.Row="1" >
<!--First tile: Schedule-->
<C4F:Tile
          Margin="1, 1, 12, 1"
          Width="180"
          Height="180"
          Title="Schedule"
          Click="Schedule_Click">
<Image Source="/icons/images/Course.png" Height="90" Width="90"
/>
</C4F:Tile>
<!--Second tile: Homework-->
<C4F:Tile
          Margin="1, 1, 12, 1"
          Width="180"
          Height="180"
          Title="Homework"
          Click="Homework_Click">
          <Image Source="/icons/images/Assignment.png"
Height="90" Width="90" />
</C4F:Tile>
</StackPanel>

<StackPanel Orientation="Horizontal" Margin="1,12,1,1"
Grid.Row="1" >
<C4F:Tile
          Margin="1, 1, 12, 1"
```

```
            Width="180" Height="180"
                Title="Notebook"
            Click="NoteBookPage_Click">
            <Image Source="/icons/images/Note.png" Height="90"
Width="90" />
</C4F:Tile>
</StackPanel>
<StackPanel Orientation="Horizontal" Margin="1,12,1,1"
Grid.Row="1">
<C4F:Tile
            Margin="1, 1, 12, 1"
            Width="180" Height="180"
                Title="About"
            Click="About_Click">
<Image Source="/icons/images/We.png" Height="90" Width="90" />
</C4F:Tile>
</StackPanel>
```

Listing 2. Code of the Main Panel

These codes make the UI when the project is running, as shown in Figure 9.

Figure 9. Main Page UI

After the main UI had been done, logical functions were added, such as when the user clicks one of the four tiles, the application redirects to a new page. The codes for logical part were added in Mainpage.xaml.cs, which is a file connected to Mainpage.xaml. A file with the .xaml suffix is used for UI design, and the file with the .xaml.cs suffix is used for logical design. These two files work with each other. Therefore, the click event must also be added to the MainPage class in Listing 3.

```
public partial class MainPage : PhoneApplicationPage.

//first: schedule tile click
private void Schedule_Click(object sender, RoutedEventArgs e)
{
```

```
            NavigationService.Navigate(new
Uri(@"/Views/CourseTimetablePage.xaml",
UriKind.RelativeOrAbsolute));
}
//second: homework tile click
private void Homework_Click(object sender, RoutedEventArgs e)
{
            NavigationService.Navigate(new
Uri(@"/Views/AssignmentBookPage.xaml",
UriKind.RelativeOrAbsolute));
}
//third: notebook tile click
private void NoteBook_Click(object sender, RoutedEventArgs e)
{
            NavigationService.Navigate(new
Uri(@"/Views/NoteBookPage.xaml", UriKind.RelativeOrAbsolute));
}
//fourth: about tile click
private void About_Click(object sender, RoutedEventArgs e)
{
            NavigationService.Navigate(new
Uri(@"/Views/About.xaml", UriKind.RelativeOrAbsolute));
}
```

Listing 3. MainPage Class Code

Finally, when the user clicks one of the four tiles, the application will redirect to the specific .xaml file.

4.2.2 Implementation of the Schedule Part

When a user clicks the schedule tile (Figure 10), he/she can see the schedule page (Figure 11).

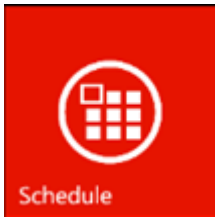

Figure 10. Schedule Tile

Figure 11. Schedule Page

The file named “CourseTimetablePage.xaml” is there to design the UI, the file “CourseTimetablePage.xaml.cs” is responsible for the logical part.

When a user clicks the plus icon, the page redirects to a new page, where the user can add detailed information about the course. The code is written in “NewOrEditCourseP-age.xaml” as shown in Figure 12.

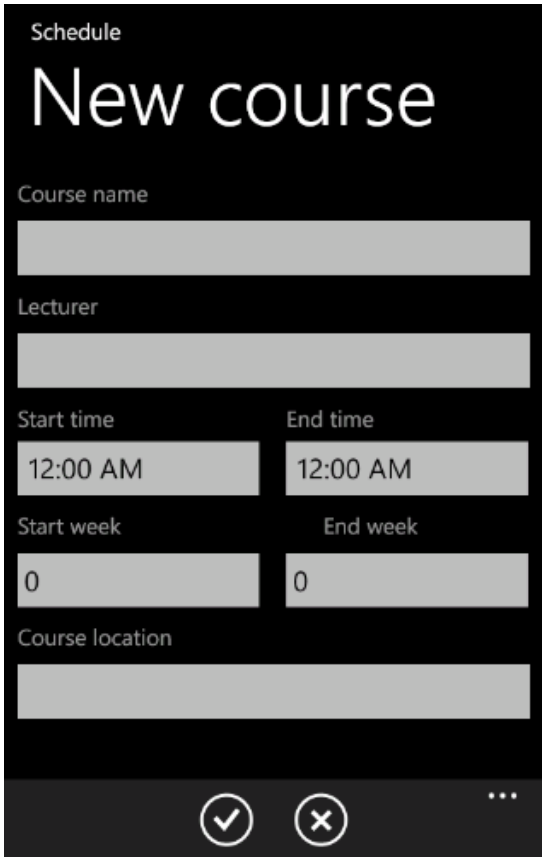

Figure 12. New Or Edit Course Page

The objects Course name, Lecturer, Start Time, End Time, Start Week, End Week, Course Location are defined in Course class, which is inherited from INotifyProperty-Changed interface [7]. INotifyPropertyChanged interface will notify clients when they change the property values.

In the “NewOrEditCoursePage.xaml” page, the Windows Phone controls Text Block, Text Box, and Time Picker are added in the file. Text Block is a control for text display, Text Box is a control for text input, and Timer Picker is a control for time input used when a user adds start time and end time. When the user clicks start time or end time, the TimePicker is shown as in Figure 13. When the user clicks the minute part (00), the view is like shown in Figure 14. Then the user can choose the minutes part. The same goes for the other parts, hour part and AM/PM part.

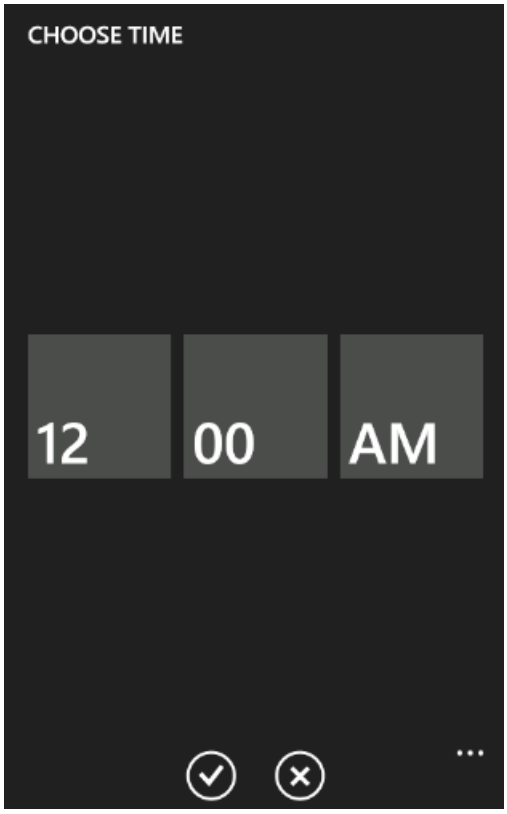

Figure 13. Time Picker

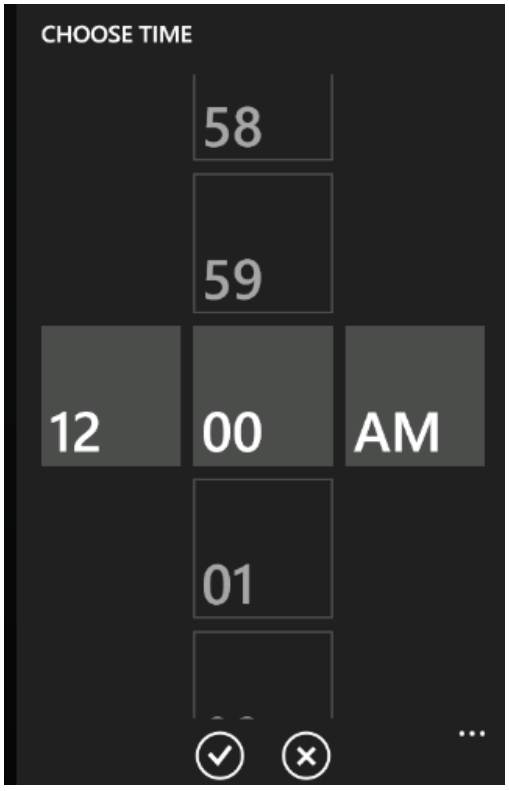

Figure 14. Time Picker.

Figure 15. Schedule Page

After the user has chosen the time and clicked the tick button, the page goes back to the “NewOrEditCoursePage.xaml” page. After all information has been entered, it is saved, and then the course information can be examined in the schedule page as seen in the Figure 15.

To achieve the function to add new courses and edit courses, INotifyPropertyChanged interface is needed in the application. The class NewOrEditCourseViewModel is inherited from the interface INotifyPropertyChanged. Two abstract methods, submit and discard, are used for saving the course information.

Once all of the above is done, implementation of the schedule part is completed.

4.2.3 Implementation of the Homework Part

The second part that needed to be implemented was the homework part. Homework should always be based on an existing course. Therefore, to create new homework for a course, the course must be added first.

First, when a user clicks the homework tile in the main UI (Figure 16), he/she can see the homework page (Figure 17).

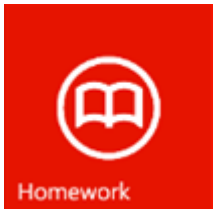

Figure 16. Homework Tile

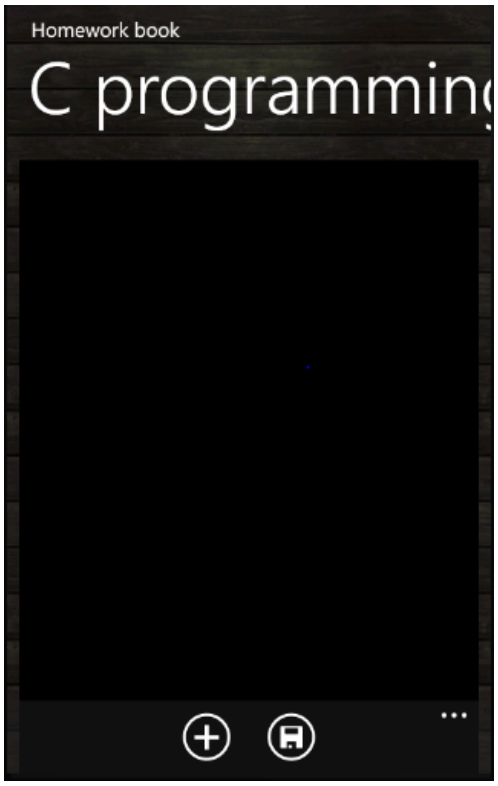

Figure 17. Homework Page

A file named “AssignmentBookPage.xaml” is created to design the UI, and “AssignmentBookPage.xaml.cs” is responsible for the logical part.

When the user clicks the plus icon, the page redirects to a new page, where the user can add detailed information about homework. The UI for this function, “NewOrEditAssignmentPage.xaml” is shown in Figure 18.

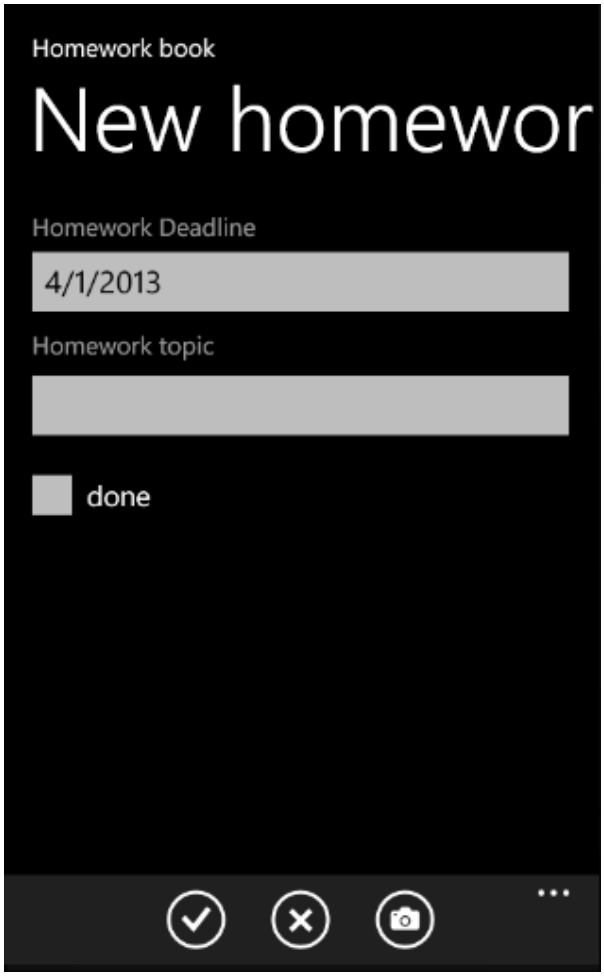

Figure 18. New Or Edit Assignment Page

Second, the class of homework must be created. The INotifyPropertyChanged interface is needed for this too. To avoid only using the INotifyPropertyChanged interface, Microsoft.Practises.Prism.dll and Microsoft.Practises.Prism.ViewModel were added to the project. Prism [8] supports a class called NotificationObject. It is used in this application. To avoid always re-declaring the INotifyPropertyChanged interface, the class NotificationObject is used, and the new class is inherited from the NotificationObject class. RaisePropertyChanged method is used to raise the object's PropertyChanged event [9].

The objects Assignment ID, Course Name, Start Time, Due Time, Content, and IsCompleted are defined in the Assignment class, which is inherited from the NotificationObject interface.

Third, the most important homework part is the data storage. In Listing 8, the interface IDataStore is created. In addition, class JsonDataStore needs to be inherited from

IDataStore. JavaScript Object Notation (JSON) is a lightweight format for interchanging data [10]. JSON serialization is used for data storage in the application [11]. Rollback and commit methods are used for data manipulation.

To show the homework to the UI part, data binding must be used. The partial code is in Listing 4.

```
<controls:Pivot Title="Homework book"
ItemsSource="{Binding AssignmentLists}" SelectedIndex="{Binding
SelectedListIndex, Mode=TwoWay}"
HeaderTemplate="{StaticResource pivotHeaderTemplate}"
ItemTemplate="{StaticResource pivotItemTemplate}"/>
<Grid.Background>
```

Listing 4. Data Binding Assignment Code

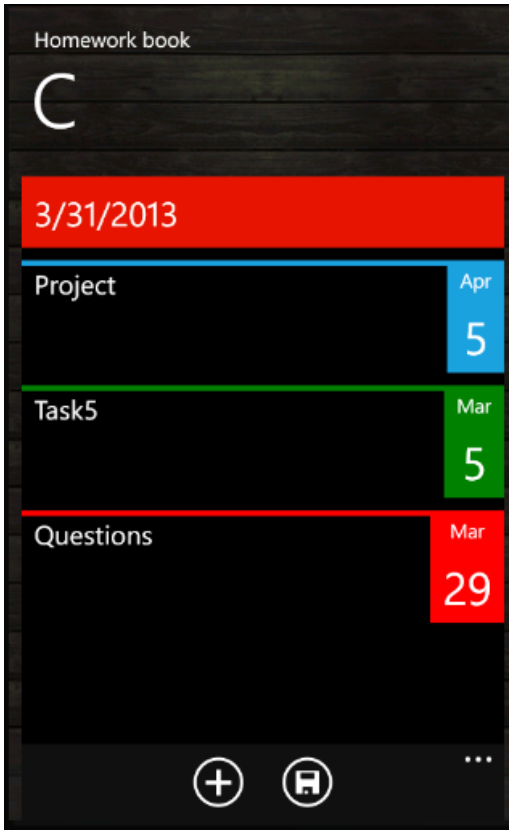

Figure 19. Homework page

Figure 19 illustrates samples of several added pieces of homework. The color of the homework depends on the state of the homework. Here are 3 situations:

1. Completed homework is shown in green.

2. Uncompleted homework is blue.
3. Overdue homework is shown in red.

Taking a simple example, in Figure 19, the blue homework “Project” means uncompleted homework with the deadline 5th of April. The green “Task5” means completed homework. The red “Questions” means overdue and uncompleted homework.

After coding, the converter is also needed to bind data, as shown in Listing 5.

```
<StackPanel>
                    <StackPanel Background="{Binding Convert-
er={StaticResource assignmentBoolToColorConverter}}" Height="5"
/>
                <TextBlock Margin="12,0" TextWrapping="Wrap"
Text="{Binding Content}" d:LayoutOverrides="Width, Height" Font-
Size="{StaticResource PhoneFontSizeMediumLarge}"/>
              </StackPanel>
               <StackPanel Grid.Column="1" Background="{Binding
Converter={StaticResource assignmentBoolToColorConverter}}" Ver-
ticalAlignment="Top">
                    <TextBlock TextWrapping="Wrap"
Text="{Binding DueDate.Month, Converter={StaticResource month-
NameConverter}, ConverterCulture=en-US}" TextAlignment="Center"
Margin="12,6" FontSize="{StaticResource PhoneFontSizeSmall}"/>
                    <TextBlock TextWrapping="Wrap"
Text="{Binding DueDate.Day}" TextAlignment="Center" Mar-
gin="12,6" FontSize="{StaticResource PhoneFontSizeExtraLarge}"/>
                </StackPanel>
```

Listing 5. Assignment to Brush Converter Data Binding

The function of color showing the homework states is successful in Figure 19.

4.2.4 Implementation of the Notebook Part

The last main function of the Student Helper is to record the course notes to the application. For this, the notebook part should be implemented. It is practical that students

who do not have pen and paper can type the notes on their mobile phones. In general, the page view looks like in Figure 20. There are two application bar buttons, the left one is for a new note and the right one is for finding tags.

Figure 20. Notebook Page View

When a user clicks the left button “new note”, he/she is redirected to the page shown in Figure 21. This is the new note page, which allows the user to enter a new note. An example of user input contents is shown in Figure 21. When the user clicks the “tick” button, the information is saved. The user can then see the information in the notebook page.

The first implementation step was to create the data structure. Thus, class note inherited from the interface NotificationObject was created. The objects Note ID, Course Name, Content, and Tag are defined in note class, which is inherited from the NotificationObject interface. These objects also use the RaisePropertyChanged method in NotificationObject.

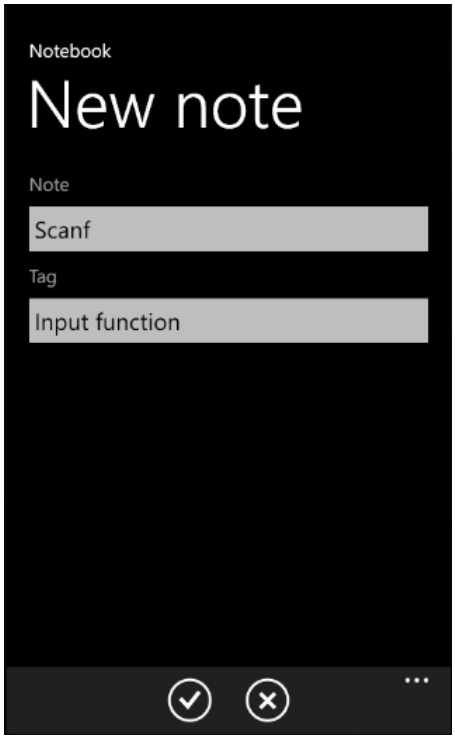

Figure 21. New Note Page View

Figure 21 shows how the page, TextBlock and TextBox appear in the application. TextBlock Code is shown in Listing 6.

```
<StackPanel>
<TextBlock TextWrapping="Wrap" Text="Note" Mar-
gin="{StaticResource PhoneHorizontalMargin}" Fore-
ground="{StaticResource PhoneSubtleBrush}"/>
<TextBox Name="NoteCourseName" Text="{Binding Item.Content,
Mode=TwoWay}" InputScope="Text">
</TextBox>
</StackPanel>
<StackPanel>
<TextBlock TextWrapping="Wrap" Text="Tag" Mar-
gin="{StaticResource PhoneHorizontalMargin}" Fore-
ground="{StaticResource PhoneSubtleBrush}"></TextBlock>
<TextBox Name="NoteContent" Text="{Binding Item.Tags,
Mode=TwoWay}" InputScope="Text">
</TextBox>
</StackPanel>
```

Listing 6. Notebook View XAML Code

Figure 22. Notebook Page View

Figure 22 displays an example of three notes added by the user. When the note “Scanf” is clicked for a long time, the application shows the view in Figure 23. Here the user can choose to edit or delete the chosen note. In the schedule part and homework part, the user can also long click the content to edit or delete.

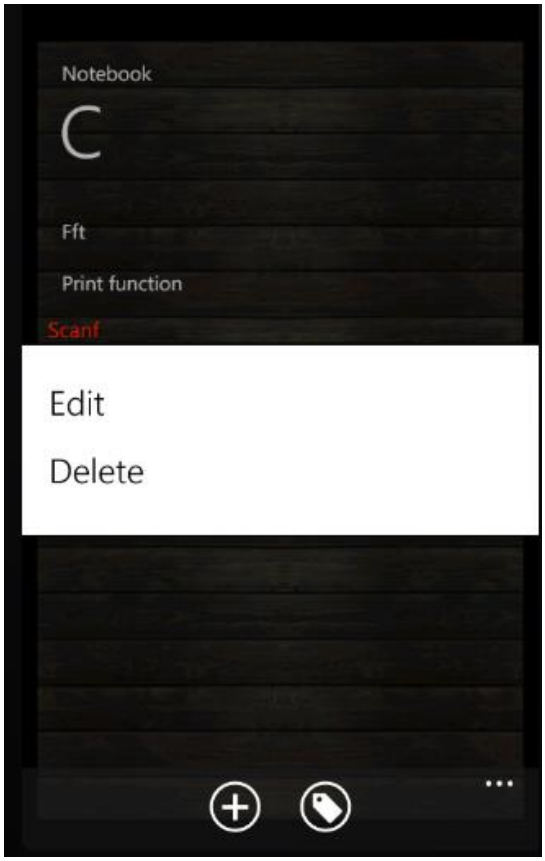

Figure 23. Edit or Delete Note View

Listing 7 shows the implementation of edit and delete methods.

```
private void EditMenuItem_Click(object sender, RoutedEventArgs e)
        {
            var menuItem = (MenuItem)sender;
            var note = (Note)menuItem.DataContext;
            NavigationService.Navigate(
                new
Uri("/Views/NewOrEditNotePage.xaml?action=edit&id=" +
                    note.Id.ToString(),
UriKind.RelativeOrAbsolute));
        }

        private void DeleteMenuItem_Click(object sender,
RoutedEventArgs e)
        {
            var menuItem = (MenuItem)sender;
```

```
            var note = (Note)menuItem.DataContext;
            App.NoteStore.Items.Remove(note);
            App.NoteStore.Commit();
        }
```

Listing 7. Code of Edit and Delete

Now the user can click the right application bar button “find tags” shown in Figure 22. It shows the tags which the user has added. The “Input function” is an example of the inputs in Figure 21. The result of showing the tags is illustrated in Figure 24.

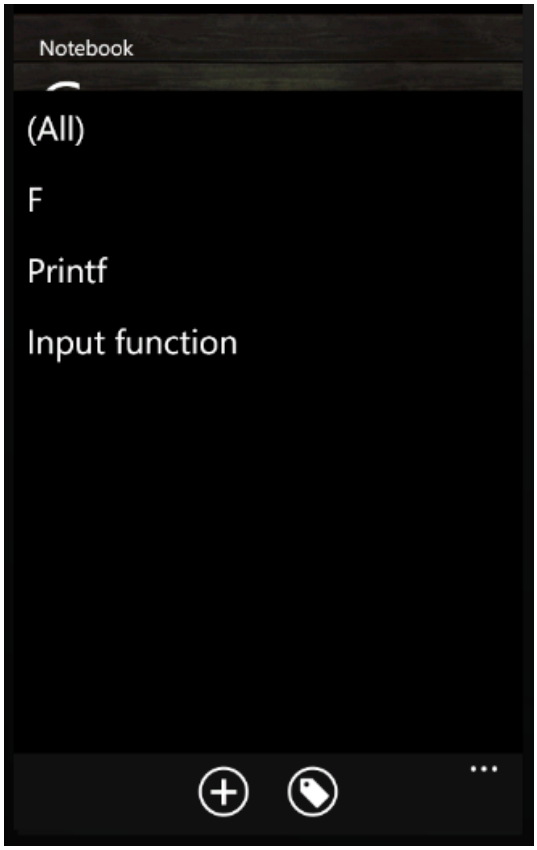

Figure 24. Find Tags Page View

In Listing 8, the code shows the implementation of showing the tags list. The user can choose a tag to find a note.

```
private void ApplicationBarTagStatictisMenuItem_Click(object
sender, System.EventArgs e)
        {
            var book = (NoteBookViewModel)DataContext;
```

```
            if (book.NoteLists.Count > 0)
            {
                var courseName = book.SelectedNoteList.Header;
                NavigationService.Navigate(
                    new Uri(
                        "/Views/TagStatisticPage.xaml?" +
                        "coursename=" + courseName,
                        UriKind.RelativeOrAbsolute));
            }
            else
            {
                MessageBox.Show("Schedule has not been created");
            }
        }
```

Listing 8. Tags display implementation

4.2.5 Implementation of the About Part

The last step is to create an about page which shows the basic information about the application. The implementation is rather easy compared to the previous three parts. Figure 24 shows the about page.

Figure 24. About Page

4.3 Testing

In general software development, testing is inevitable. There is no doubt that bugs exist so testing is very important. Testing is done to show how the program performs and to find the flaws in the application before it is used. When the program is tested, the application is executed by using artificial data. Testing is done to detect errors, bugs, abnormalities or exceptions in the program as well. [13,206]

To test this application, the testing procedure was as follows:

1. Run the application in the Windows phone emulator.
2. If it runs normally without errors, click the first tile, Schedule, to test its functions.
3. Add some courses, and check whether the results are displayed normally or not.
4. Edit course information, and then check if the results are displayed normally or not.
5. Go back to the main page, and then click the second tile, Homework, to test its functions.
6. Add some homework, and check if the results are displayed normally or not.
7. Edit homework information, and then check if the results are displayed normally or not.
8. Go back to the main page, and then click the third tile, Notebook, to test its functions.
9. Add some notes, and check whether the results are displayed normally or not.
10. Edit the notebook information, and then check if the results are displayed normally or not.
11. Go back to the main page, and then clicking the last tile, About, to test its functions.

If everything is OK, the application is assumed to function correctly.

After the testing procedures, the application can be deployed in a real mobile phone. When the application is deployed in a real phone, the testing procedures need to be repeated on the phone. If every procedure runs successfully, the testing part is completed. Additionally, the application was tested by other students for a trial. The users recommended an improved UI. However, the users noticed the Student Helper functions well and without bugs.