

现代操作系统应用开发实验报告

学号： 15331046

班级： 软工 1 班

姓名： 陈志扬

实验名称： HW6-Todos

一．参考资料

请在这里列出对本实验有帮助你所参考的资料或者网站。

资料：老师上课所提供的课件、TA 师兄的作业要求 ppt

网站：[文件、文件夹和库](#)、[SQLite 数据库](#)、<http://stackoverflow.com/>
等等

二．实验步骤

请在这里简要写下你的实验过程。

- ① 首先学习老师提供的 PPT。主要学习文件、文件夹和库的处理，理解 LocalFolder、RoamingFolder 的作用；另外，本次实验最重要的学会 SQLite 数据库的使用，理解并使用数据库的四大基本功能“增删改查”。LocalFolder、RoamingFolder 和 StringBuilder 的作用在心得体会中体现。
- ② 根据作业要求，我以 HW3 的作业为 demo，开始我的 HW6 作业之旅。按照作业 ppt 的指导，在 vs 中安装 SQLite，建立数据库和表，然后逐步实现数据库的“增删改查”。
- ③ “增”：当 create 一个新 item 时，在数据库中增加对应的数

据

```
,  
// 将NewItem增加到数据库中  
private void insertTodo(string title, string description, string time)  
{
```

④ “删”：当 delete 一个 item 时，在数据库中删除相应的数据

```
// 删除数据库对应的Item  
private void deleteTodo(string title, string description)  
{
```

⑤ “改”：当 update 一个 item 时，在数据库中更新对应的数据

```
// 更新数据库中对应的Item  
private void updateTodo(string title, string description)  
{
```

⑥ “查”：先在 MainPage 上设置控件 Textbox 和 Button，添加 Click 为

SearchButton_Clicked，当点击 Search 时，进行查询并显示所查到的信息。

```
// 点击Search按钮，进行数据库的查询  
private void SearchButton_Clicked(object sender, RoutedEventArgs e)  
  
// 根据一个字符串模糊搜索匹配的Item并返回  
private List<TodoItem> GetTodo(string searchText)
```

⑦ 实现从数据库中读取数据并显示在 MainPage 中，具体代码在第四部分中体现

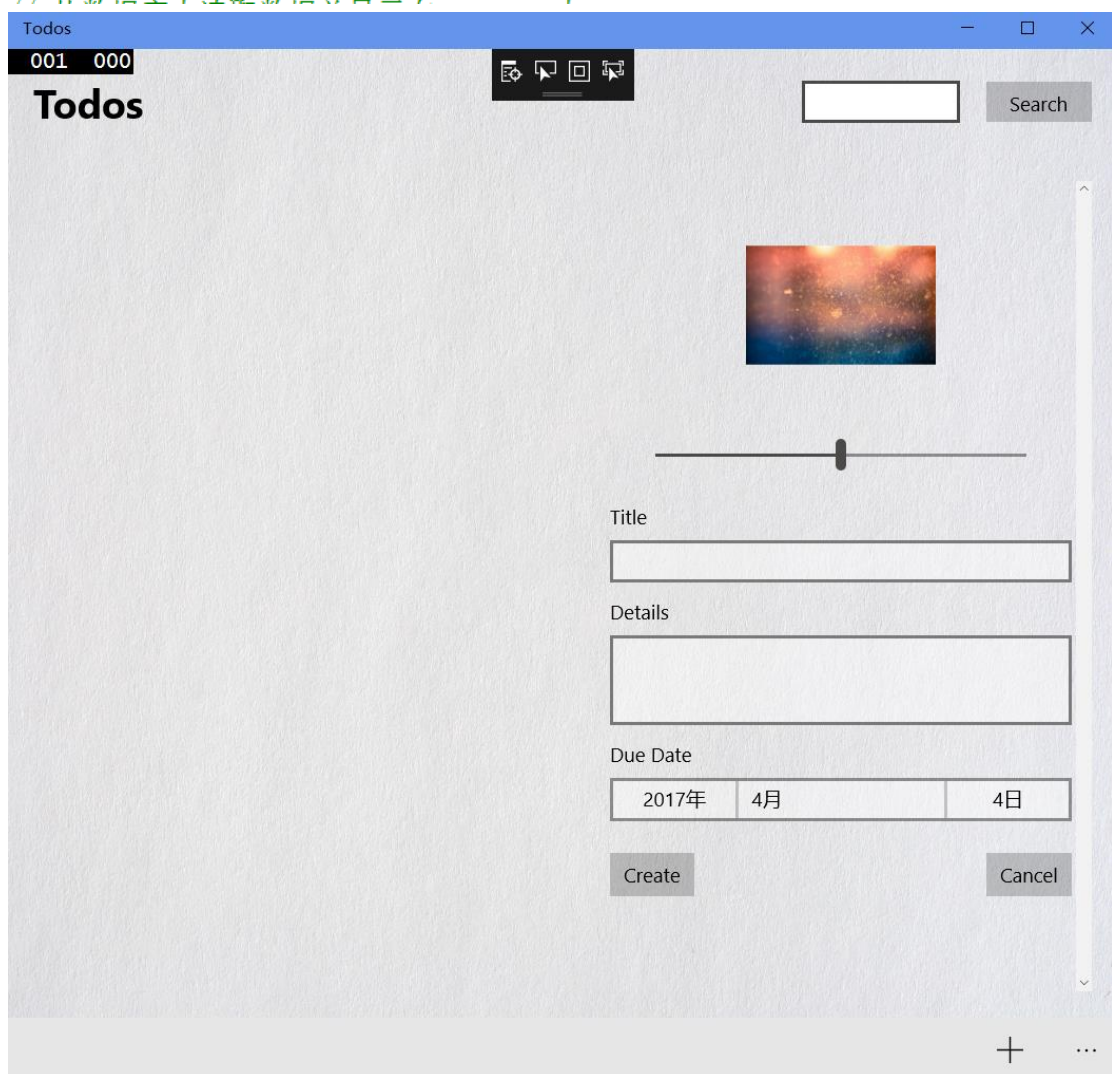
⑧ 最后，在 NewPage 中实现增加、删除、修改的代码。

三．实验结果截图

请在这里把实验所得的运行结果截图。

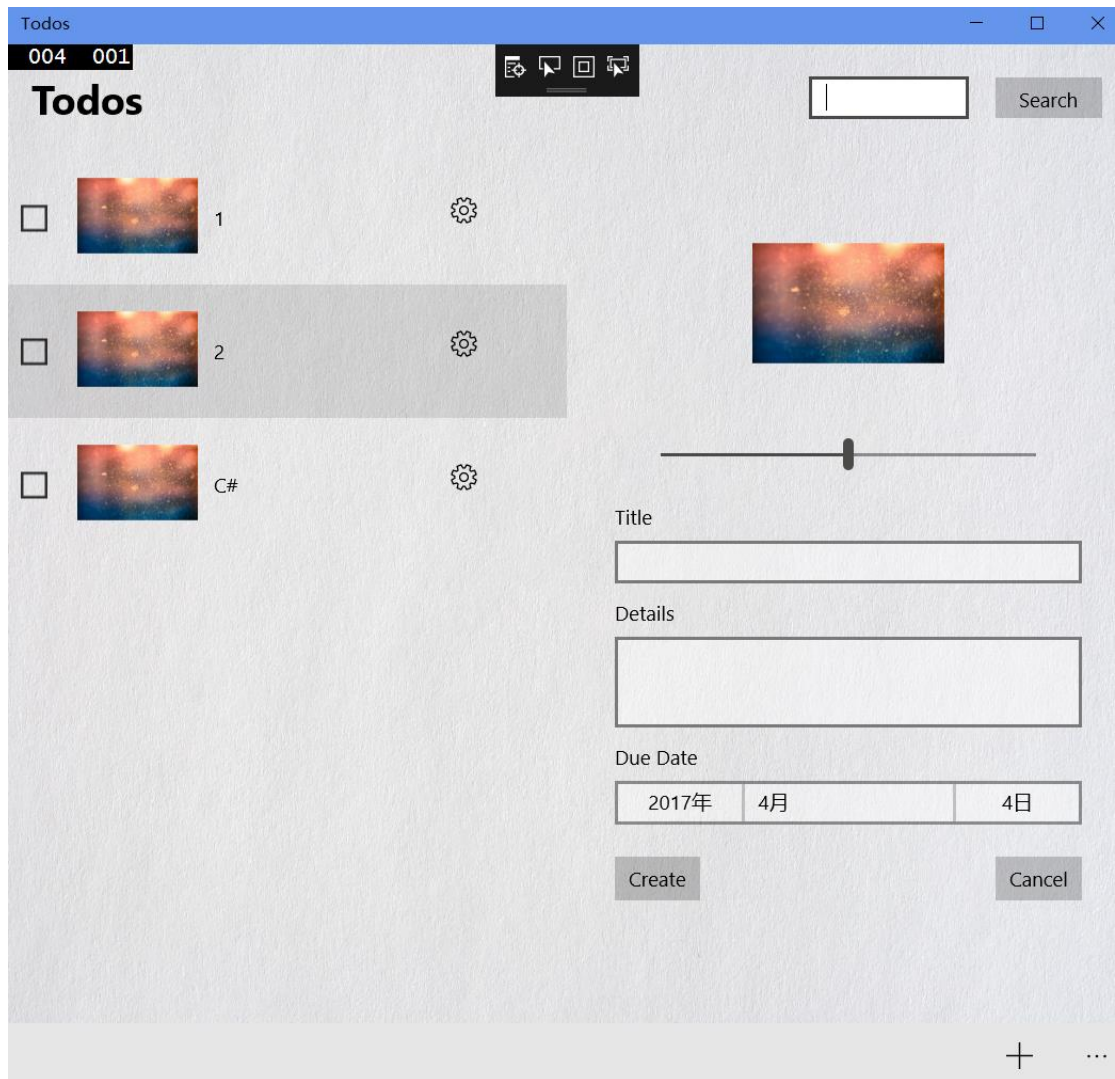
① 程序启动：由于我把代码中这部分注释掉，所以现在没有一个 item

```
//this.allItems.Add(new Models.TODOItem("123", "123", DateTime.Now.Date));
//this.allItems.Add(new Models.TODOItem("456", "456", DateTime.Now.Date));
```

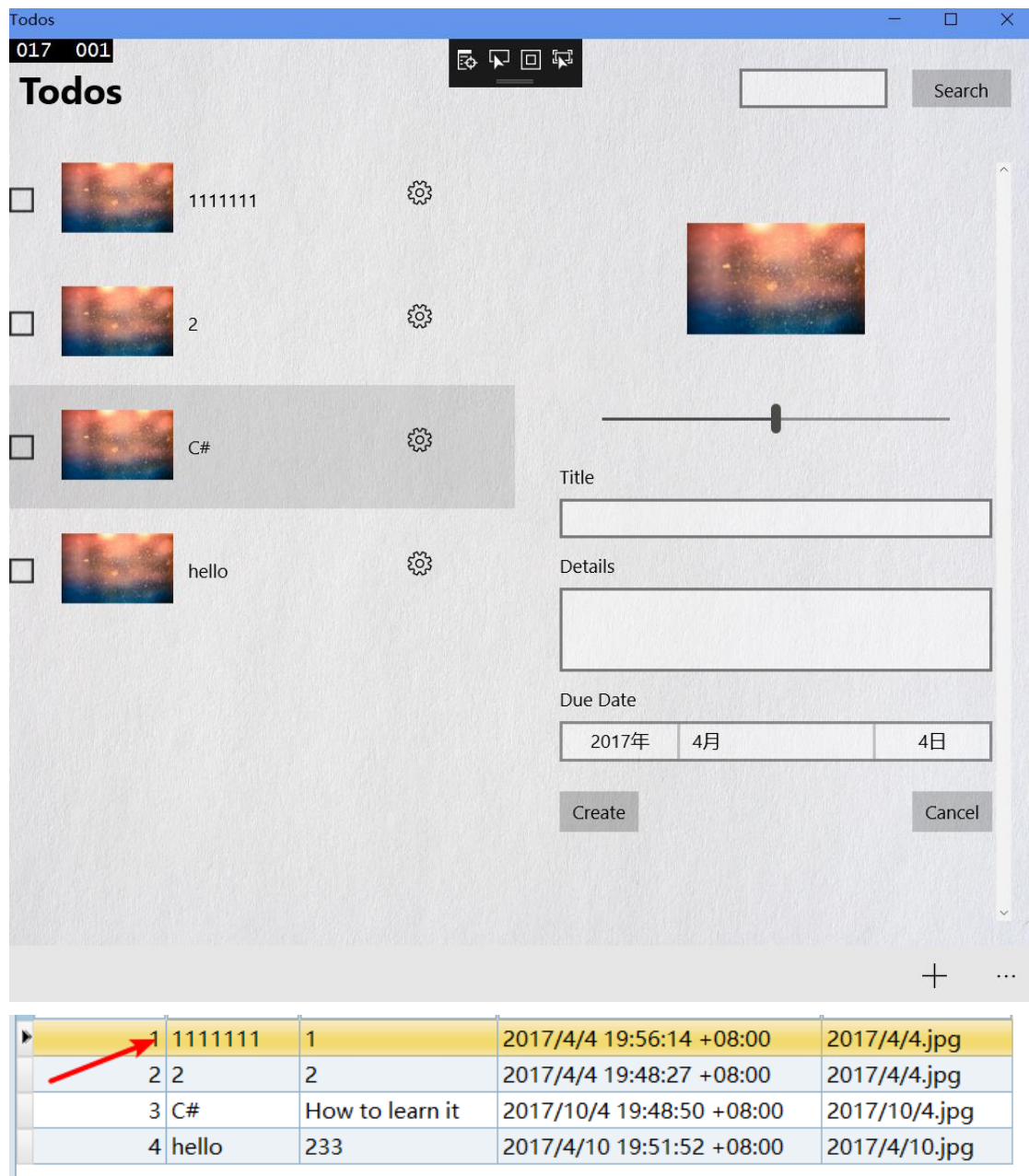


② 在数据库中新建如下的数据，重新启动程序，可见此时成功从数据库中读取数据并显示在 MainPage 中

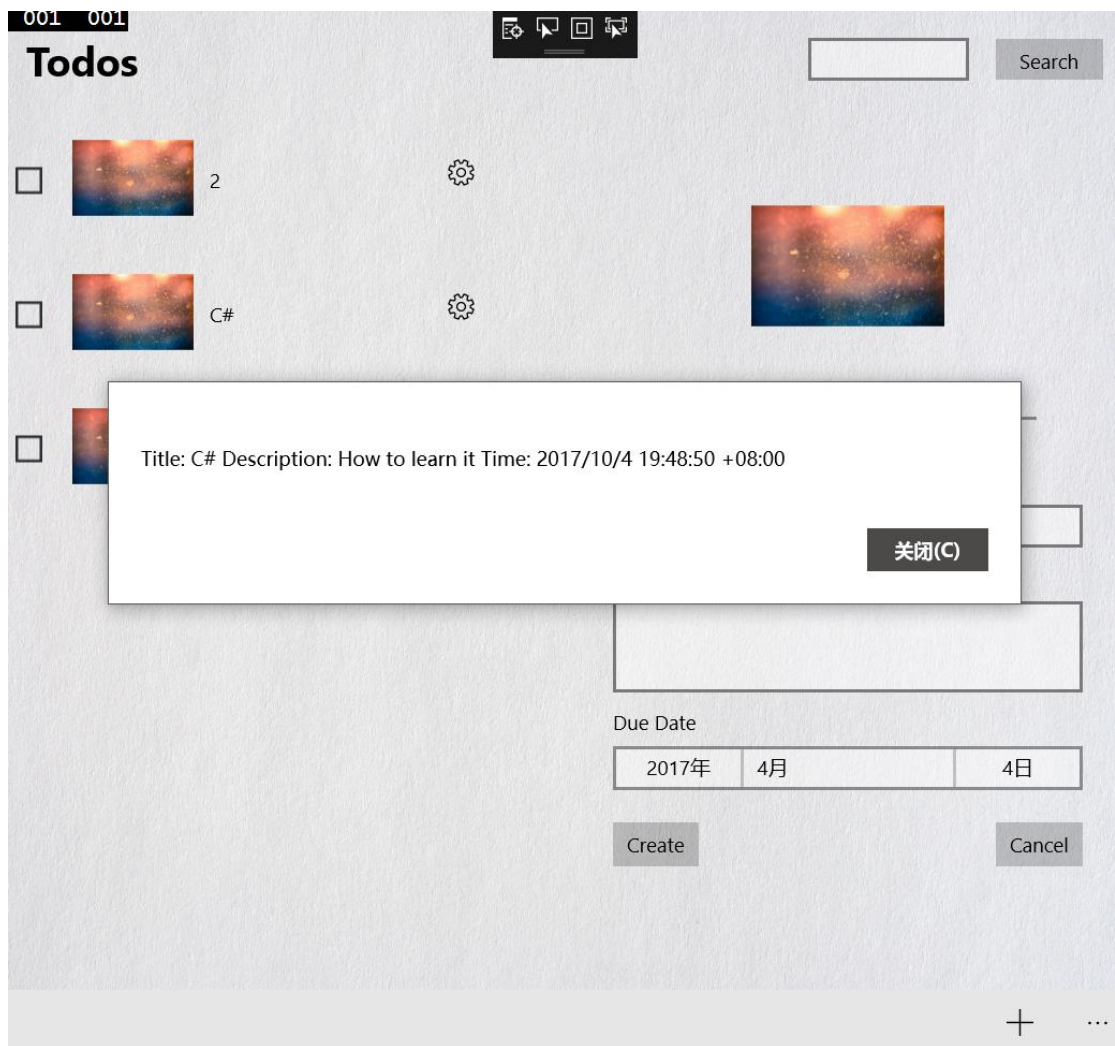
rowid	Title	Description	Time	Picture
(empty)	(empty)	(empty)	(empty)	(empty)
1	1	1	2017/4/4 19:43:56 +08:00	2017/4/4.jpg
2	2	2	2017/4/4 19:48:27 +08:00	2017/4/4.jpg
3	C#	How to learn it	2017/10/4 19:48:50 +08:00	2017/10/4.jpg




- ③ 在 MainPage 中新建一个 Item，在数据库中可见新增加了一行对应的数据





- ⑤ 在 MainPage 中也可以直接删除一个 item , 点击 Setting 中的 Delete 可执行该功能 , 在数据库中可以看到对应的数据已被删除 , 例如删除第一个 item (title 为 1111111)





另外，可以新建几个数据进行查询，以显示查询得到的多个 item，例如在新建如下的数据后，再搜索框中输入 2 进行查询：


☐

2
⚙️


☐

C#
⚙️

☐

hello
⚙️

☐

2
⚙️

☐

245
⚙️

☐

524
⚙️



Title

Details

Due Date

2017年

4月

4日

Create

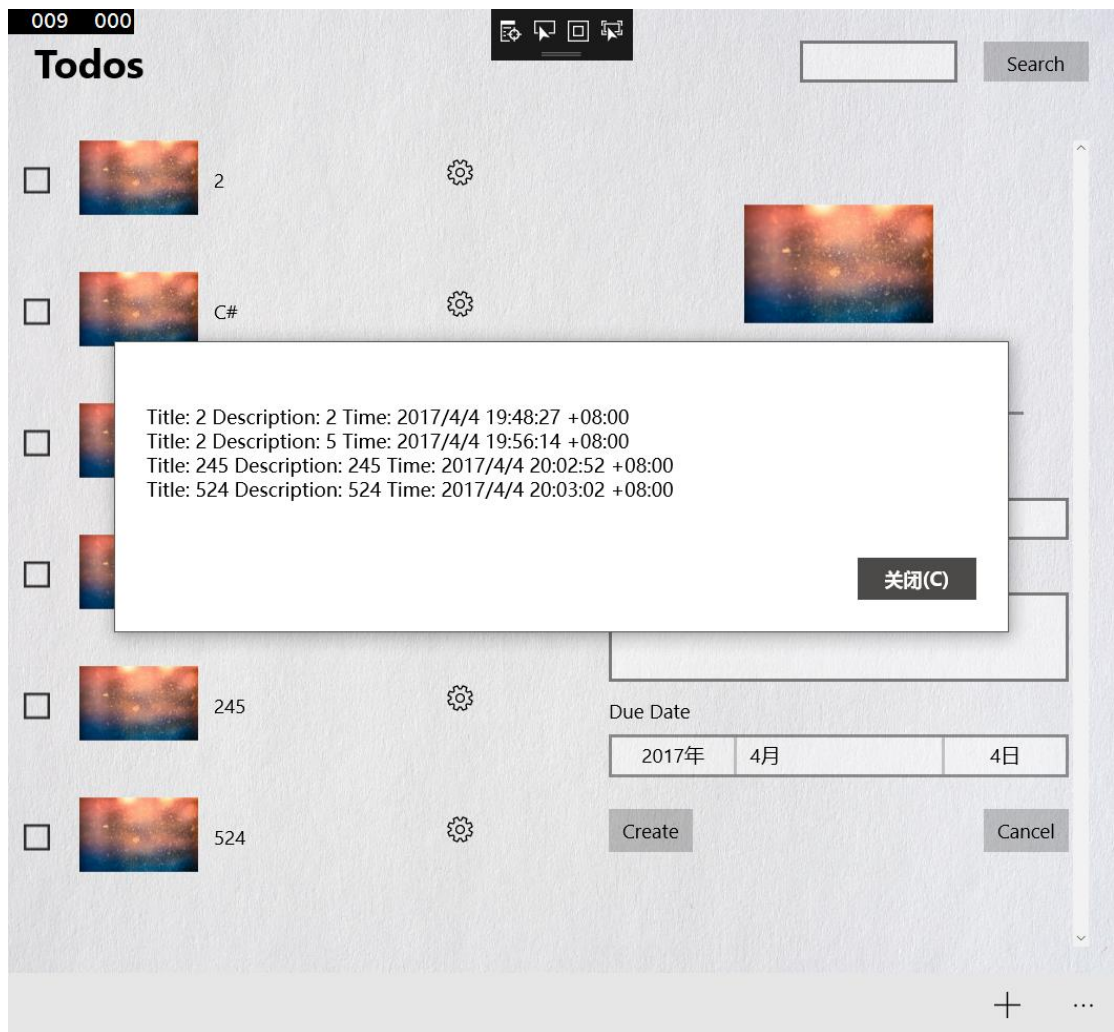
Cancel

➤	➔2	2	2	2017/4/4 19:48:27 +08:00	2017/4/4.jpg
	3	C#	How to learn it	2017/10/4 19:48:50 +08:00	2017/10/4.jpg
	4	hello	233	2017/4/10 19:51:52 +08:00	2017/4/10.jpg
	➔5	2	5	2017/4/4 19:56:14 +08:00	2017/4/4.jpg
	➔6	245	245	2017/4/4 20:02:52 +08:00	2017/4/4.jpg
	➔7	524	524	2017/4/4 20:03:02 +08:00	2017/4/4.jpg

2

×





Search



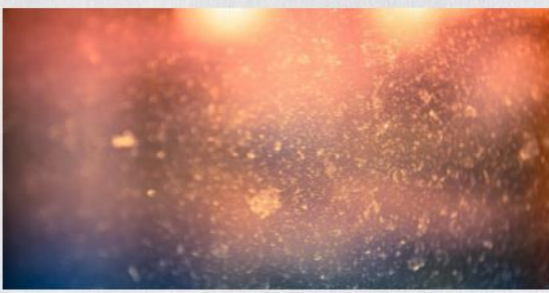
⑦ 在 NewPage 中创建新 item , 数据库中也能增加一行对应的数据


- ⑧ 在窄屏下点击一个 item 跳转到 NewPage 中，进行 update，在数据库中也可见对应的数据被更新，例如点击 title 为 C# 的 item，将其更新为 C++，如下：

003 000

Edit Todo Item



 select

Title

C++


Details


How to learn it

Due Date

2017年	10月	4日
-------	-----	----

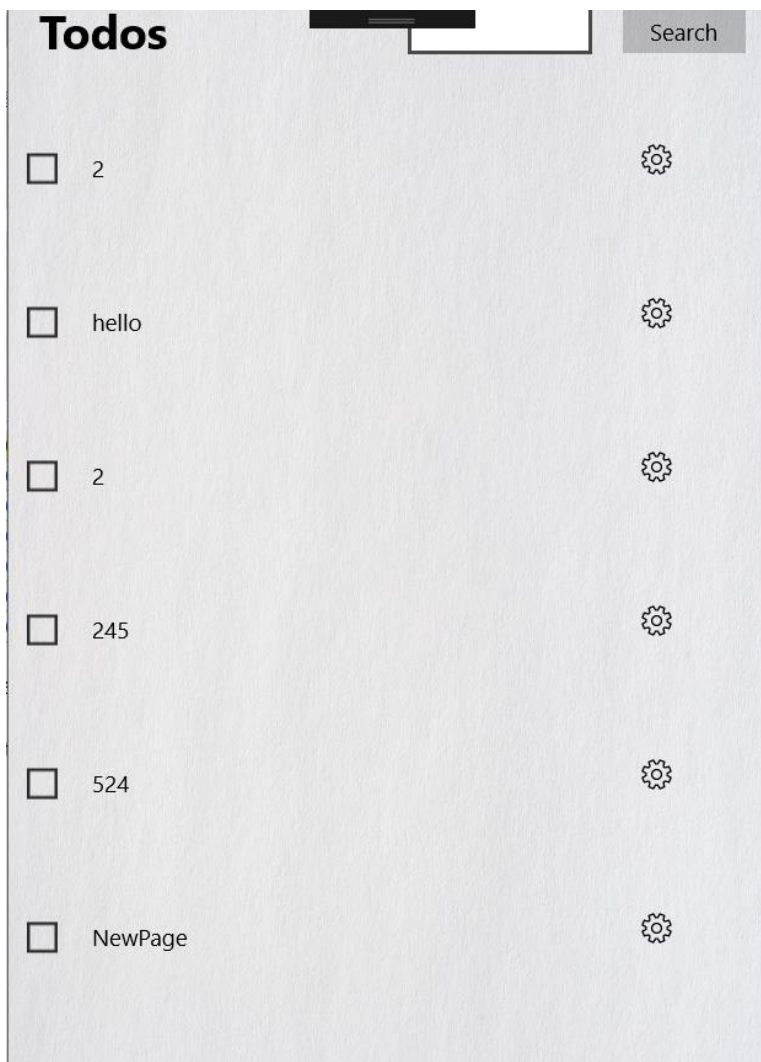
Update Cancel

 ...

☐ C++ 

2	2	2	2017/4/4 19:48:27 +08:00	2017/4/4.jpg
3	C++	How to learn it	2017/10/4 20:12:33 +08:00	2017/10/4.jpg
4	hello	233	2017/4/10 19:51:52 +08:00	2017/4/10.jpg
5	2	5	2017/4/4 19:56:14 +08:00	2017/4/4.jpg
6	245	245	2017/4/4 20:02:52 +08:00	2017/4/4.jpg
7	524	524	2017/4/4 20:03:02 +08:00	2017/4/4.jpg
8	NewPage	NewPage	2019/4/4 20:07:20 +08:00	2019/4/4.jpg

⑨ 在窄屏下点击一个 item 跳转到 NewPage 中，进行删除，在数据库中可见对应的数据也被删除，例如，点击 title 为 C++ 的 item，并把它删除：



2	2	2	2017/4/4 19:48:27 +08:00	2017/4/4.jpg
4	hello	233	2017/4/10 19:51:52 +08:00	2017/4/10.jpg
5	2	5	2017/4/4 19:56:14 +08:00	2017/4/4.jpg
6	245	245	2017/4/4 20:02:52 +08:00	2017/4/4.jpg
7	524	524	2017/4/4 20:03:02 +08:00	2017/4/4.jpg
8	NewPage	NewPage	2019/4/4 20:07:20 +08:00	2019/4/4.jpg

四．实验过程遇到的问题

请在这里写下你在实验过程中遇到的问题以及解决方案。

- ① 最主要的问题还是图片。因为之前没做图片的绑定，这次做到后面才知道要有绑定图片才能对图片进行存取，但是一切都太晚了，需要在 TodoItem 类中增加 BitmapImage 和 Uri 才能实现图片的绑定，然后 TodoItem 类和 TodoItemViewModel 类中的各个函数也需要修改，工作量巨大，再尝试做了一个下午后还是放弃了，所以图片这方面没做，我只是简单把图片用“时间选取器中的日期+.jpg”显示在数据库中。虽然如此，但我已经知道怎么实现图片绑定和图片路径保存到数据库中。有机会有时间一定会将其实现。
- ② 从数据库中读取数据并在 MainPage 中显示，ppt 中 SQLiteResult.DONE == statement.Step()总是不对，经过断点调试发现根本不等于，改成 ROW 即可。

```

public TodoItemViewModel()
{
    // 加入两个用来测试的item
    //this.allItems.Add(new Models.TodoItem("123", "123", DateTime.Now.Date));
    //this.allItems.Add(new Models.TodoItem("456", "456", DateTime.Now.Date));
    // 从数据库中读取数据并显示在MainPage中
    using (var db = new SQLiteConnection("Todos.db"))
    {
        using (var statement = db.Prepare("SELECT Title, Description, Time FROM Todo"))
        {
            while (SQLiteResult.ROW == statement.Step())
            {
                this.allItems.Add(new Models.TodoItem((string)statement[0], (string)statement[1],
                    DateTimeOffset.Parse(statement[2].ToString())));
            }
        }
    }
}

```

- ③ 一开始创建数据库表总是出现语法错误,主要原因还是不熟悉数据库的语言,缺个逗号、括号、双引号都很正常。下图是我所创建的数据库 Todos 和表 Todo

```

private void LoadDatabase()
{
    // 创建数据库Todos, 表Todo
    var conn = new SQLiteConnection("Todos.db");
    string sql = @"CREATE TABLE IF NOT EXISTS
                    Todo(Title VARCHAR(40),
                        Description VARCHAR(40),
                        Time VARCHAR(40),
                        Picture VARCHAR(40));";
    using (var statement = conn.Prepare(sql))
    {
        statement.Step();
    }
}

```

- ④ 向数据库中增加数据库时,由于 DatePicker.Date 需要转换成字符串,所以用到 ToString()方法

```

// 将NewItem插入到数据库中
insertTodo(Title.Text, Details.Text, DatePicker.Date.ToString());

```

```
// 将NewItem增加到数据库中
private void insertTodo(string title, string description, string time)
{
    using (var db = new SQLiteConnection("Todos.db"))
    {
        using (var temp = db.Prepare("INSERT INTO Todo(Title, Description, Time, Picture) VALUES(?, ?, ?, ?)"))
        {
            temp.Bind(1, title);
            temp.Bind(2, description);
            temp.Bind(3, time);
            temp.Bind(4, DatePicker.Date.ToString("d") + ".jpg");
            temp.Step();
        }
    }
}
```

⑤ 修改 item 时，日期的处理是个大问题，要考虑到月份和日期

有两位数的可能，一开始没考虑到这个问题，只是这样写：截取日期的前八个字符，再加上当前时间的时分秒，这样就忽略了两位数的问题，导致查询功能出错。

```
temp.Bind(3, DatePicker.Date.ToString().Substring(0, 9) + DateTime.Now.ToString().Substring(9));
```

进行以下的修改即可：

```
// 时间选取器上的日期+当前时间(时分秒)
temp.Bind(3, DatePicker.Date.ToString("d") + " " + DateTime.Now.TimeOfDay.ToString().Substring(0, 8) + " +08:00");
```

完整的 updateTodo 代码如下：

```
// 更新数据库中对应的Item
private void updateTodo(string title, string description)
{
    using (var db = new SQLiteConnection("Todos.db"))
    {
        using (var temp = db.Prepare("UPDATE Todo SET Title = ?, Description = ?, Time = ?, Picture = ? WHERE Title = ? AND Description = ?"))
        {
            temp.Bind(1, Title.Text);
            temp.Bind(2, Details.Text);
            // 时间选取器上的日期+当前时间(时分秒)
            temp.Bind(3, DatePicker.Date.ToString("d") + " " + DateTime.Now.TimeOfDay.ToString().Substring(0, 8) + " +08:00");
            temp.Bind(4, DatePicker.Date.ToString("d") + ".jpg");
            temp.Bind(5, title);
            temp.Bind(6, description);
            temp.Step();
        }
    }
}
```


⑥ 接下来是查询问题。主要有以什么形式保存查到的数据，数据库的查询语句 Like 和 % 的使用这两个问题。关于第一点，可以用 List<TodoItem> 来保存；第二点仔细想想数据库的语言即可解决。完整查询代码：

```
// 根据一个字符串模糊搜索匹配的Item并返回
private List<TodoItem> GetTodo(string searchText)
{
    List<TodoItem> todos = new List<TodoItem>();
    TodoItem class Todos.Models.TodoItem
    if (V
    {
        if (searchText != "")
        {
            using (var db = new SQLiteConnection("Todos.db"))
            {
                using (var statement = db.Prepare("SELECT Title, Description, Time FROM Todo WHERE Title LIKE ?"))
                {
                    statement.Bind(1, "%" + searchText + "%");
                    while (SQLiteResult.ROW == statement.Step())
                    {
                        todo = new TodoItem((string)statement[0], (string)statement[1], DateTimeOffset.Parse(statement[2].ToString()));
                        todos.Add(todo); // 将查询到的todo添加到todos结尾处
                    }
                }
            }
        }
    }
    return todos;
}
```

⑦ 如何把查询得到的结果显示出来。根据提示可用 StringBuilder 及其 AppendLine 方法

```

// 点击Search按钮，进行数据库的查询
private void SearchButton_Clicked(object sender, RoutedEventArgs e)
{
    // GetTodo获取查询到的Item
    var todos = GetTodo(search.Text);
    // 构建StringBuilder类的一个实例message，用来显示查询到的信息
    StringBuilder message = new StringBuilder();
    if (todos != null)
    {
        if (todos.Count > 0)
        {
            for (int i = 0; i < todos.Count; i++)
            {
                // 将信息追加到message中
                message.AppendLine("Title: " + todos[i].title + " Description: " +
                    todos[i].description + " Time: " + todos[i].day);
            }
            // 弹出MessageBox，显示信息
            var j = new MessageBox(message.ToString()).ShowAsync();
        }
        else
        {
            var j = new MessageBox("There is no such Todo in the list.").ShowAsync();
        }
    }
    search.Text = "";
}

```

⑧ 删除功能比较容易，简单带过。

```

// 删除数据库对应的Item
private void deleteTodo(string title, string description)
{
    using (var db = new SQLiteConnection("Todos.db"))
    {
        using (var statement = db.Prepare("DELETE FROM Todo WHERE Title = ? AND Description = ?"))
        {
            statement.Bind(1, title);
            statement.Bind(2, description);
            statement.Step();
        }
    }
}

```

五．思考与总结

请在这里写下你本次试验的心得体会以及所思所想。

心得体会：

LocalFolder：获取本地应用程序数据存储区中的根文件夹，可以

访问本地应用程序数据存储区的文件。每个应用程序都有一个存储数据的文件夹，一般来说程序只能访问自己的文件夹，对其中的文件可读可写。

RoamingFolder：获取漫游应用程序数据存储区中的根文件夹，然而必须遵守同步引擎对文件名约定的限制以确保漫游文件夹中的项可以漫游，同步引擎也会限制漫游的设置和文件的大小。具体来说，如果用户在一台设备上安装使用了应用程序，在另一台设备上使用该应用程序时应该把可漫游的项传送到这台设备上供用户使用。RoamingFolder 存放漫游的项。

StringBuilder：因为 string 对象是不可变的，每次使用 string 都需要在内存中创建一个新的 string 对象，浪费内存空间。因此，在需要对字符串执行重复修改的情况下，我们采用 StringBuilder 可以显著提高性能。

这次实验真的是体现了项目设计框架的重要性。由于之前没实现图片的绑定，导致这次想添加该功能时代码修改量巨大，可谓“牵一发而动全身”。经过这次教训，吸取经验，在期中项目可以避开很多坑，应该仔细思考整个设计框架，按照设计文档需求来实现，不能中途乱改，应时刻考虑整体框架。

另外，也深刻体会到这样的名言“程序不是没有 bug，而是还没发现 bug”。

1. 实验报告提交格式为 pdf。

2. 实验内容不允许抄袭，我们要进行代码相似度对比。如发现抄袭，按 0 分处理。