- University of zakho
- Department of computer science
- Faculty of science
- Subject: - Mobile Application
- Semester – 8 –

Academic Year: 2021/2022

# REPORT ABOUT

# "TODO APPLICATION"

**Prepared By:-**

- Saraa ahmad mahmud
- Harem Shaheen Omer
- Jalila Zaki ahmad
- Alnd Wahed Ahmad

**Lecturers name:-Mr.Yousif Garabet**

**Group – B -**

## Table of Contents pages

## 1. Introduction

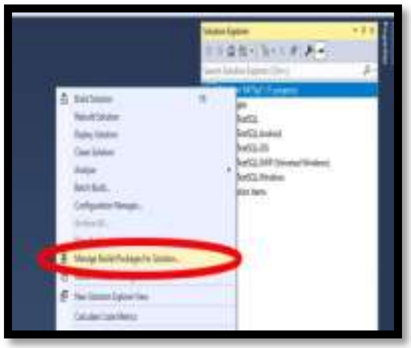
The name of our app is Todo App it is used for writing notes so if the user needs to write any note he will use it easily and this app has some process that will apply it at the note that will make easy to use this Todo App.

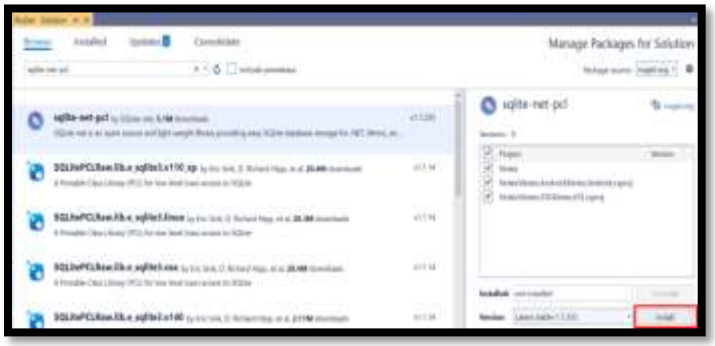
- **Xamarin.Forms**

  Xamarin. Forms is an open source cross-platform framework from Microsoft for building iOS, Android, & Windows apps with . NET from a single shared codebase. Use Xamarin. Forms built in pages, layouts, and controls to build and design mobile apps from a single API that is highly extensible.

- **SQLite**

  SQLite is an open source database, available on every android database. It supports standard relations database features, like SQL syntax, transactions & SQL statements. SQLite is considerably, the lighter version of SQL database, where most of the SQL commands don't run on SQLite database, is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. It is a database, which is zero-configured, which means like other databases you do not need to configure it in your system.





- **Install Sqlite**

➢ Launch Visual Studio and open a solution.

➢ In Solution Explorer, right-click the solution and select Manage NuGet Packages for Solution...

- **ToDo App:**

To develop our app we used Xamarine .forms and SQLite to do the following:

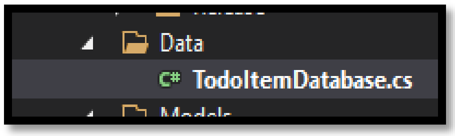
1- create the first page to be able the user write the name of note and can make save ,delete ,cancel , speak so those function the user will have.

2- to create the second page we are have button to the right ion top that when we are clicked to it user will  have new note.

## 2.App Design And Implementation

### 2.1 Database Design:

To create our app we first needed to create a local database to save and delete  , canel, and speak  Todo app information, for that purpose we created 1 class file is called TodoItemDatabase. cs which will be the database.

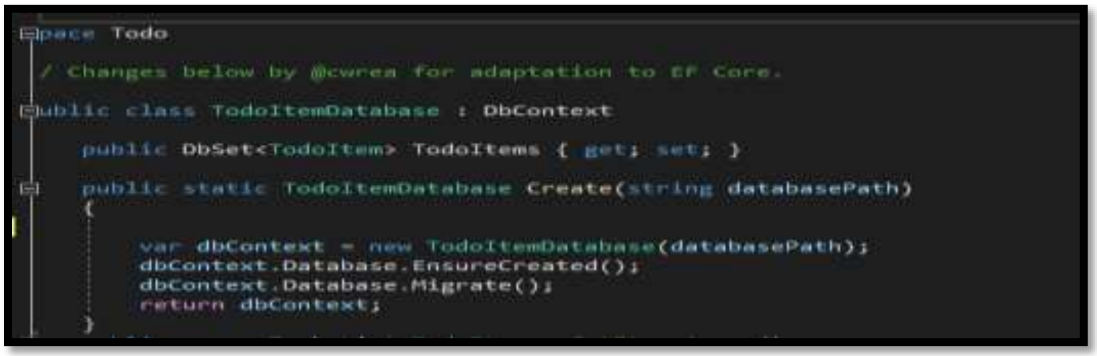


```
Data
    TodoItemDatabase.cs
Models
```

### 2.2 Database code:

#### 1. Constructor

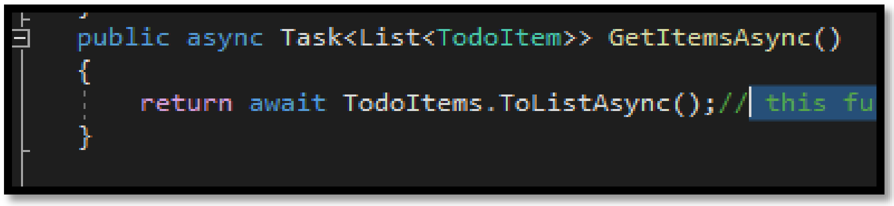
Create a database in the path that was set, if there is a database already created retrieve the created database, if there is no database create a new database in the path given



```
pace Todo

/ Changes below by @cwrea for adaptation to EF Core.

ublic class TodoItemDatabase : DbContext

    public DbSet<TodoItem> TodoItems { get; set; }

    public static TodoItemDatabase Create(string databasePath)
    {

        var dbContext = new TodoItemDatabase(databasePath);
        dbContext.Database.EnsureCreated();
        dbContext.Database.Migrate();
        return dbContext;
    }
```

## 2.List Function

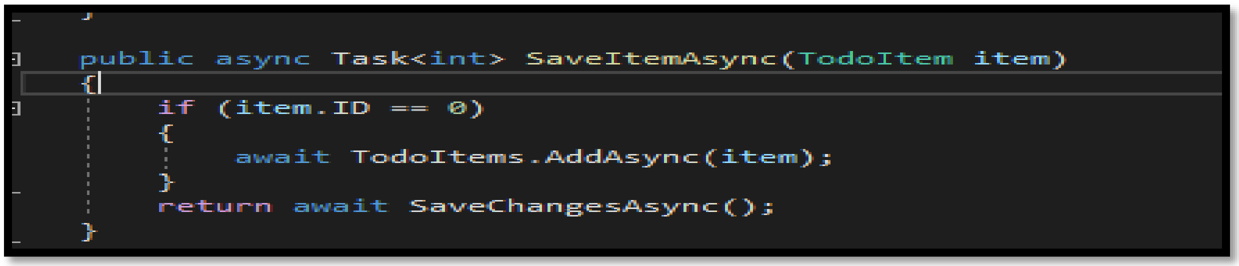
this function returns Todo item table and convert it to list



```
public async Task<List<TodoItem>> GetItemsAsync()
{
    return await TodoItems.ToListAsync();// this fu
}
```

## 3.Save Function

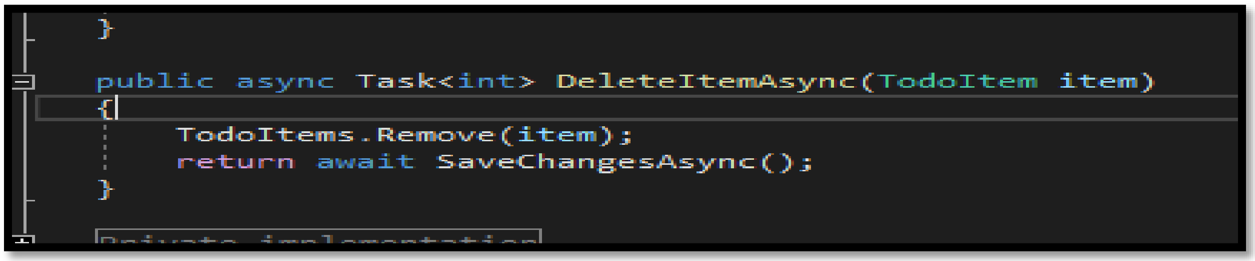
this function save data to todo item table using TodoItem Model class and save it to the database.



```
public async Task<int> SaveItemAsync(TodoItem item)
{
    if (item.ID == 0)
    {
        await TodoItems.AddAsync(item);
    }
    return await SaveChangesAsync();
}
```

## 4.Delete Function

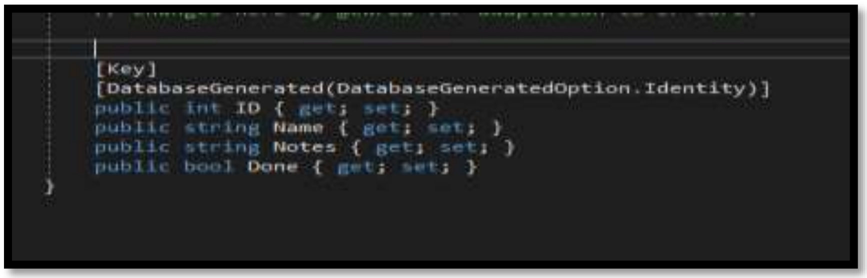
this function delete data from TodoItem table using the TodoItem Model class and delete it from the database



```
}

public async Task<int> DeleteItemAsync(TodoItem item)
{
    TodoItems.Remove(item);
    return await SaveChangesAsync();
}
```

## 5.Database Creation

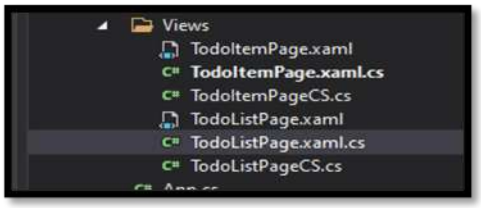
TodoItem table is declared here and its attributes are defined.



```
    [Key]
    [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    public int ID { get; set; }
    public string Name { get; set; }
    public string Notes { get; set; }
    public bool Done { get; set; }
}
```

## 2.3 Content Pages  Design:

After creating the database now we need to create several pages for the user to interact with, our Content Page files:

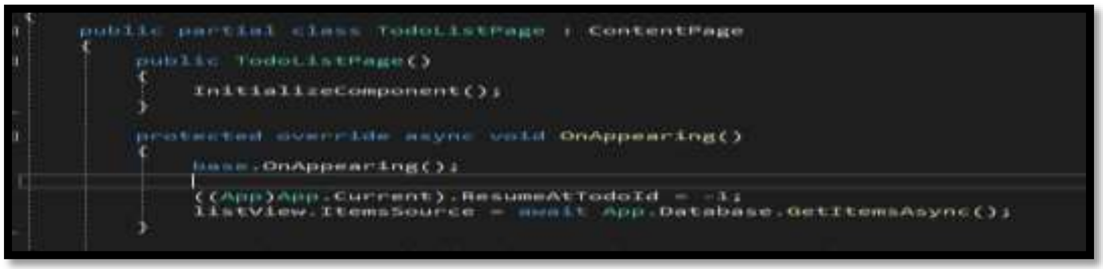


```
◢ 🗁 Views
    TodoItemPage.xaml
    TodoItemPage.xaml.cs
    TodoItemPageCS.cs
    TodoListPage.xaml
    TodoListPage.xaml.cs
    TodoListPageCS.cs
```

## 2.4 Content Pages Code:

### On Appearing

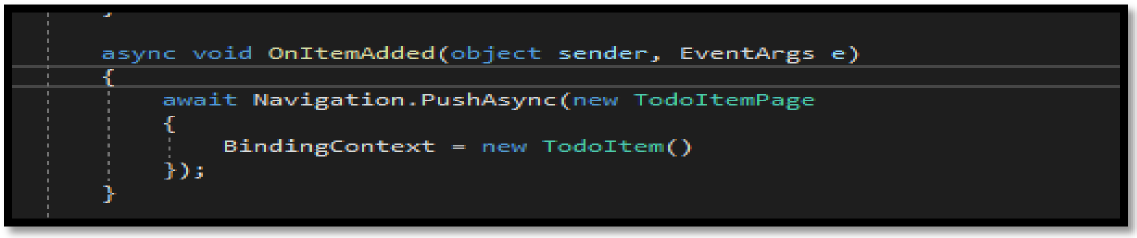
when the page appeared it will get the items from the database and set it to the list view .



```
{
    public partial class TodoListPage : ContentPage
    {
        public TodoListPage()
        {
            InitializeComponent();
        }

        protected override async void OnAppearing()
        {
            base.OnAppearing();

            ((App)App.Current).ResumeAtTodoId = -1;
            listView.ItemsSource = await App.Database.GetItemsAsync();
        }
```

### On Item Added

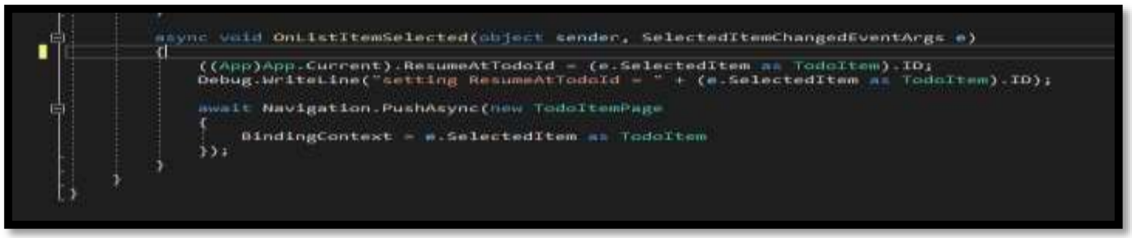
plus button it will go to the page TodoItempage



```
    async void OnItemAdded(object sender, EventArgs e)
    {
        await Navigation.PushAsync(new TodoItemPage
        {
            BindingContext = new TodoItem()
        });
    }
```

### OnListItemSelected

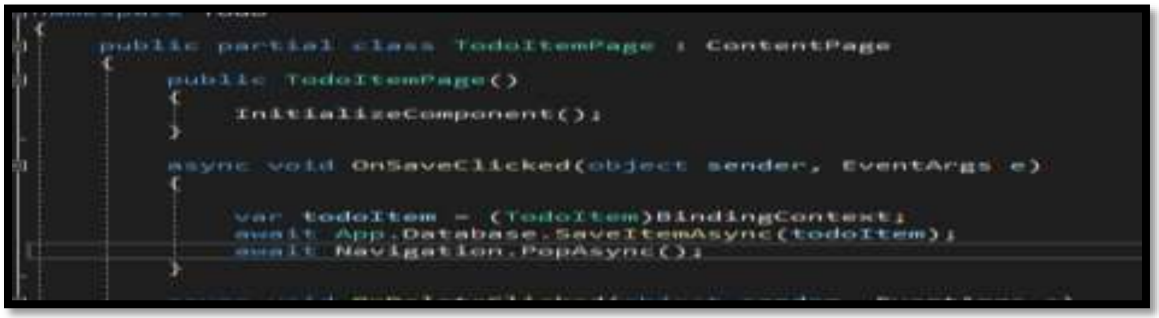
when this function is called it will go to the TodoListPage and it will set the entry values to the values given.



```
        async void OnListItemSelected(object sender, SelectedItemChangedEventArgs e)
        {
            ((App)App.Current).ResumeAtTodoId = (e.SelectedItem as TodoItem).ID;
            Debug.WriteLine("setting ResumeAtTodoId = " + (e.SelectedItem as TodoItem).ID);

            await Navigation.PushAsync(new TodoItemPage
            {
                BindingContext = e.SelectedItem as TodoItem
            });
        }
    }
}
```

## OnSaveClicked Event

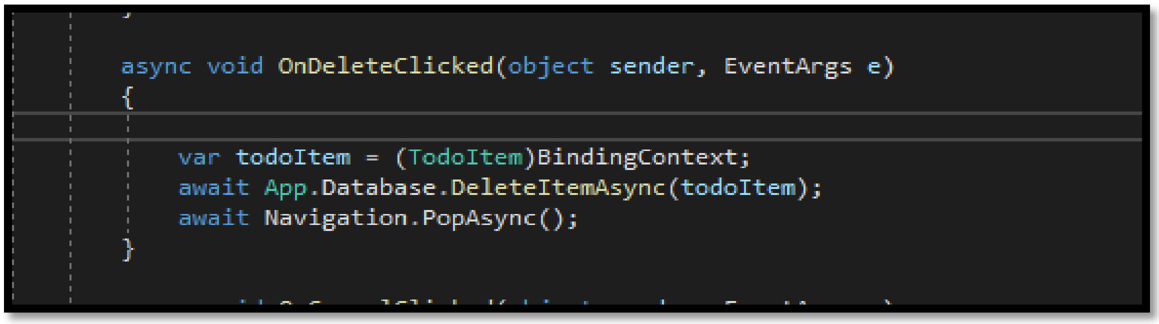
it is used to take input from the user and save it  to the TdoItem table in the database and pop the page and go back to the previous page



```
    public partial class TodoItemPage : ContentPage
    {
        public TodoItemPage()
        {
            InitializeComponent();
        }

        async void OnSaveClicked(object sender, EventArgs e)
        {

            var todoItem = (TodoItem)BindingContext;
            await App.Database.SaveItemAsync(todoItem);
            await Navigation.PopAsync();
        }
```

## OnDeleteClicked Event

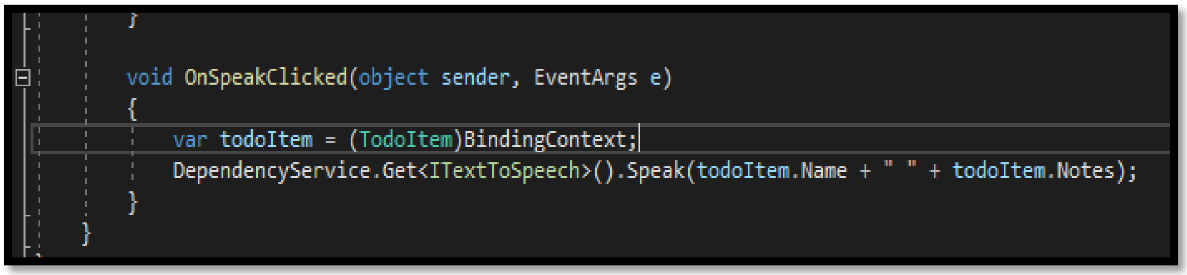
is used to get the selected item and delete it from the database
 and pop the page and go back to the previous page,PopAsync()



```
        async void OnDeleteClicked(object sender, EventArgs e)
        {

            var todoItem = (TodoItem)BindingContext;
            await App.Database.DeleteItemAsync(todoItem);
            await Navigation.PopAsync();
        }
```

## OnSpeakClicked  Event

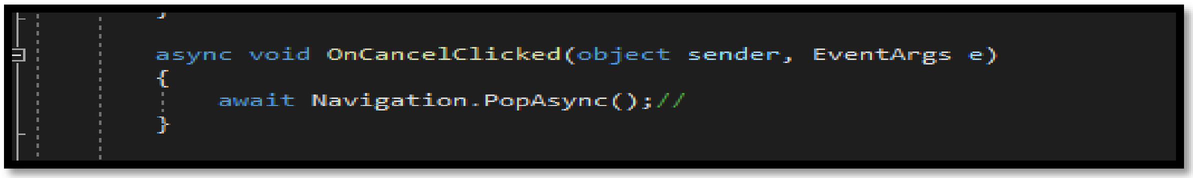
take the input from the user and make the app speak those inputs when we are writing it in the app.



```
        }

        void OnSpeakClicked(object sender, EventArgs e)
        {
            var todoItem = (TodoItem)BindingContext;
            DependencyService.Get<ITextToSpeech>().Speak(todoItem.Name + " " + todoItem.Notes);
        }
    }
```

### OnCancelClicked Event

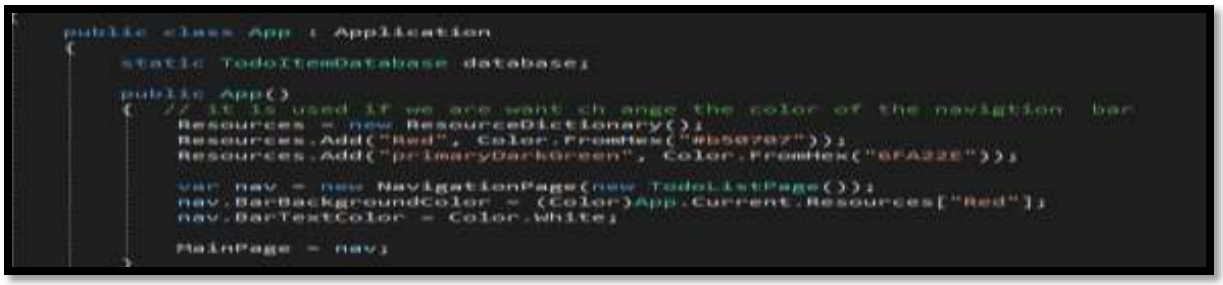
When this function is called the current page will be popped and it will go back to the previous page



```
async void OnCancelClicked(object sender, EventArgs e)
{
    await Navigation.PopAsync();//
}
```

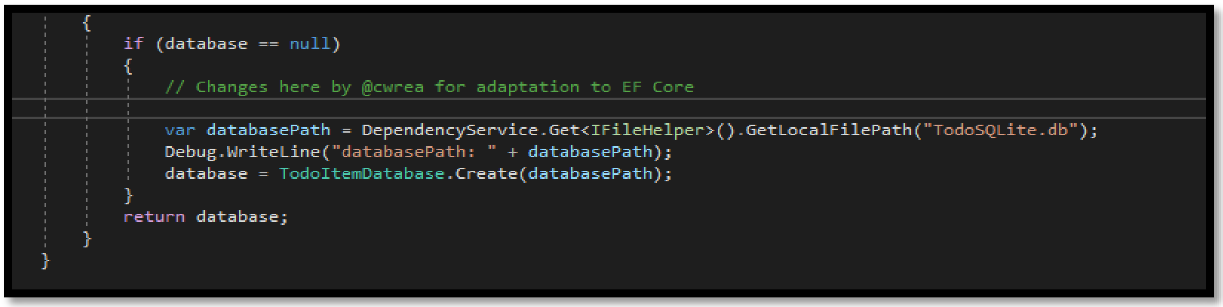
## 2.5 App.cs

it is used if we are want to change the color of  the navigation  bar



```
public class App : Application
{
    static TodoItemDatabase database;

    public App()
    {   // it is used if we are want ch ange the color of the navigtion  bar
        Resources = new ResourceDictionary();
        Resources.Add("Red", Color.FromHex("#b50707"));
        Resources.Add("primaryDarkGreen", Color.FromHex("6FA22E"));

        var nav = new NavigationPage(new TodoListPage());
        nav.BarBackgroundColor = (Color)App.Current.Resources["Red"];
        nav.BarTextColor = Color.White;

        MainPage = nav;
    }
```

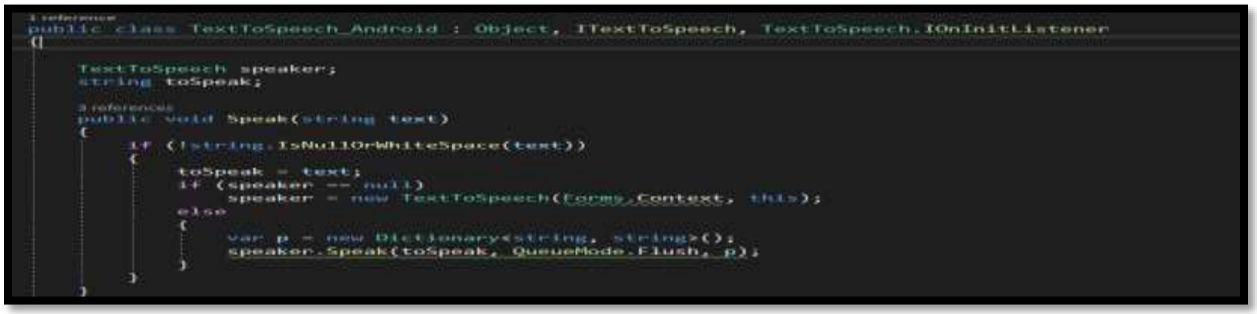
create a static variable that is accessible globally from the other classes



```
        {
            if (database == null)
            {
                // Changes here by @cwrea for adaptation to EF Core

                var databasePath = DependencyService.Get<IFileHelper>().GetLocalFilePath("TodoSQLite.db");
                Debug.WriteLine("databasePath: " + databasePath);
                database = TodoItemDatabase.Create(databasePath);
            }
            return database;
        }
    }
```

### TextToSpeach

this class is providing the TextToSPEACH service to the application



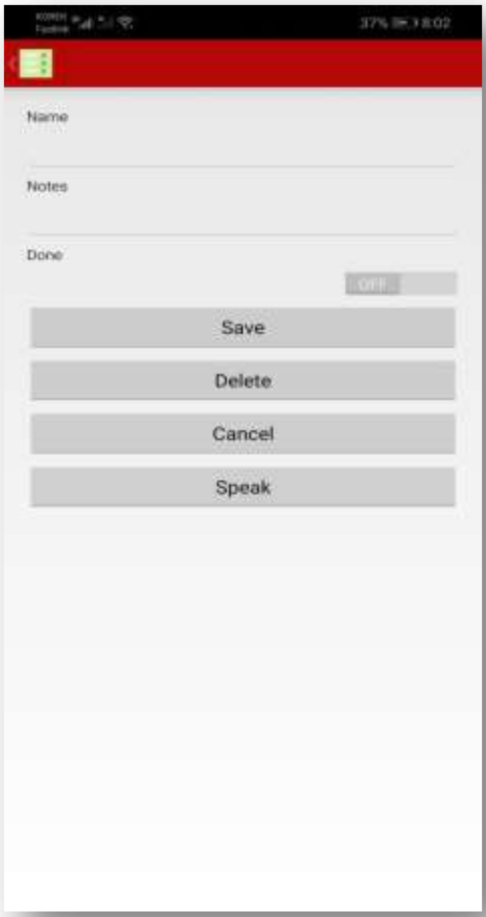
```
1 reference
public class TextToSpeech_Android : Object, ITextToSpeech, TextToSpeech.IOnInitListener
{
    TextToSpeech speaker;
    string toSpeak;

    3 references
    public void Speak(string text)
    {
        if (!string.IsNullOrWhiteSpace(text))
        {
            toSpeak = text;
            if (speaker == null)
                speaker = new TextToSpeech(Forms.Context, this);
            else
            {
                var p = new Dictionary<string, string>();
                speaker.Speak(toSpeak, QueueMode.Flush, p);
            }
        }
    }
```

# 3.APPLICATION UI

## 3.1 Application Icons

*page for list*

This is first page and to the right on the top we are have the plus icon it is used if we are want to add new note every time we want add just by clicking to the icon it will go to the second page and at the second page we will write the name of note and number of Todo.

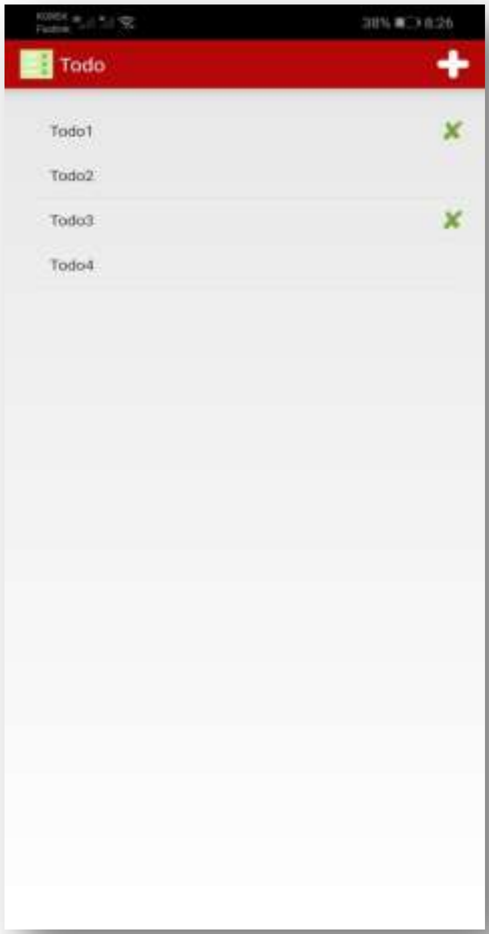


*second page for to do process for the note*

Here the user has four buttons if the user wants to make some process to his note from the update and delete button and the user if want out of the app will clicking to the cancel button. Another button is a speak button it is used to read the name and note with sound.

The switch above it is used when user has finished write his note so the finish mark will appear front note at the first page

## 3.1 Output



Todo1

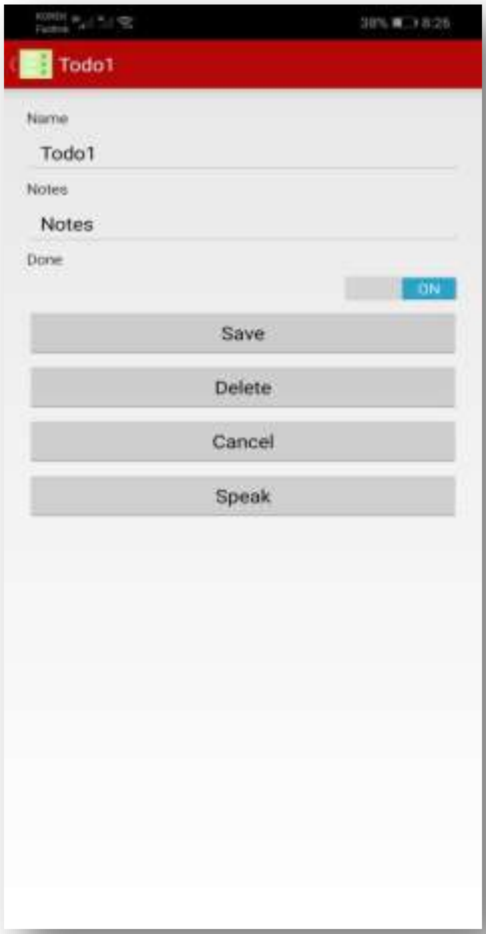
Todo2

Todo3

Todo4

*page after insert note*

Here we are having the list of notes that user wrote it so the whole note will display here at this first page.



*page after write input*

after the user write the name of note and write his note like above in screen.

---