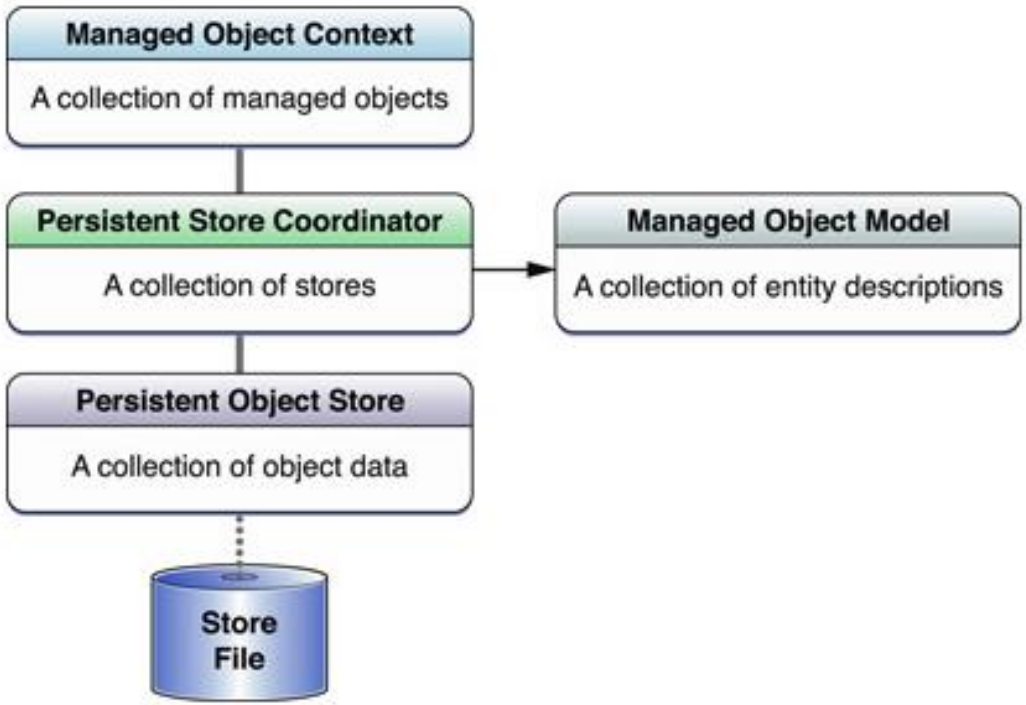
iPhone Apps Development using Objective C & Xcode (Lesson 7)

By Dannis Mok

Core Data Stack

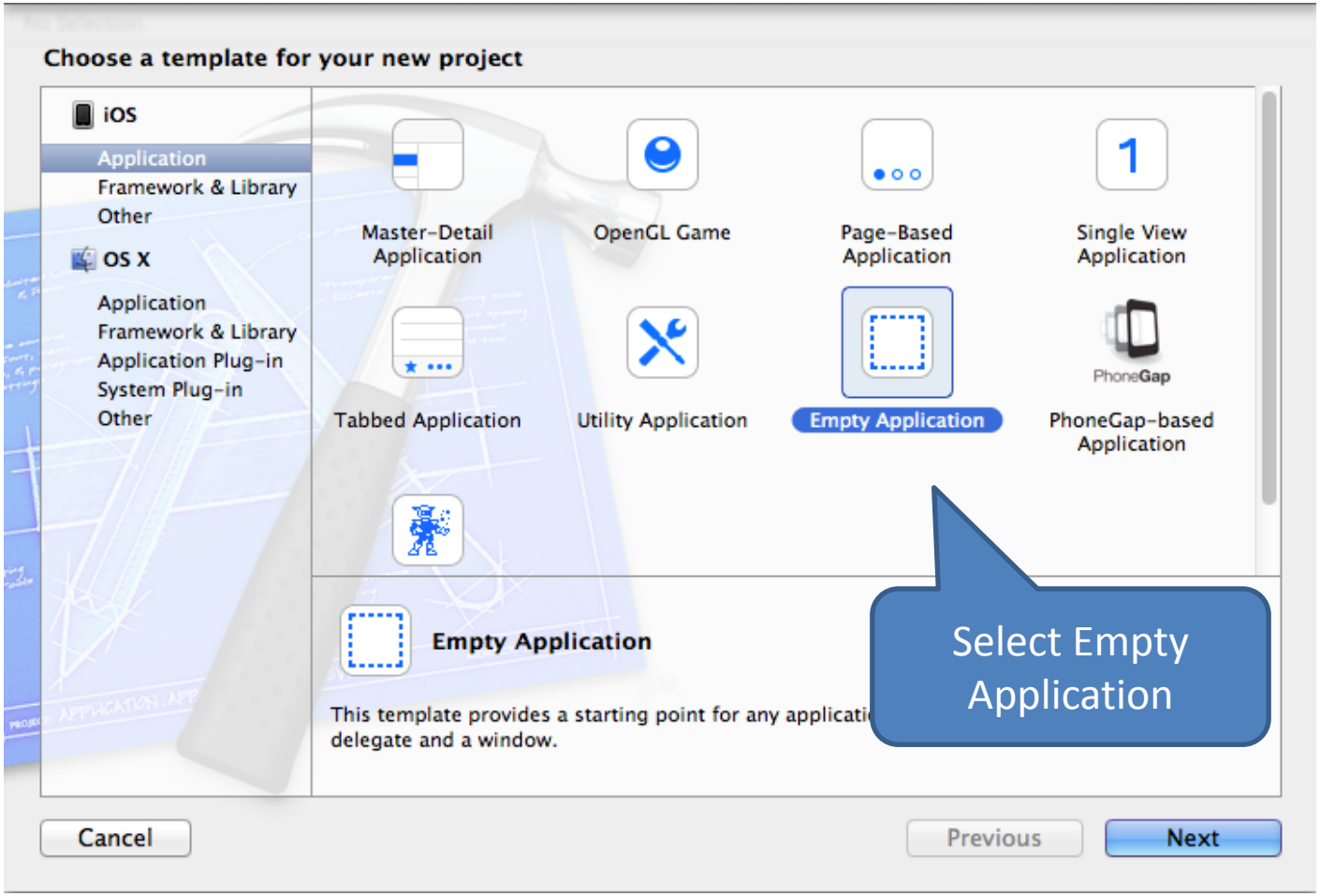
- A Core Data stack is composed of the following objects: one or more managed object contexts connected to a single persistent store coordinator which is in turn connected to one or more persistent stores.



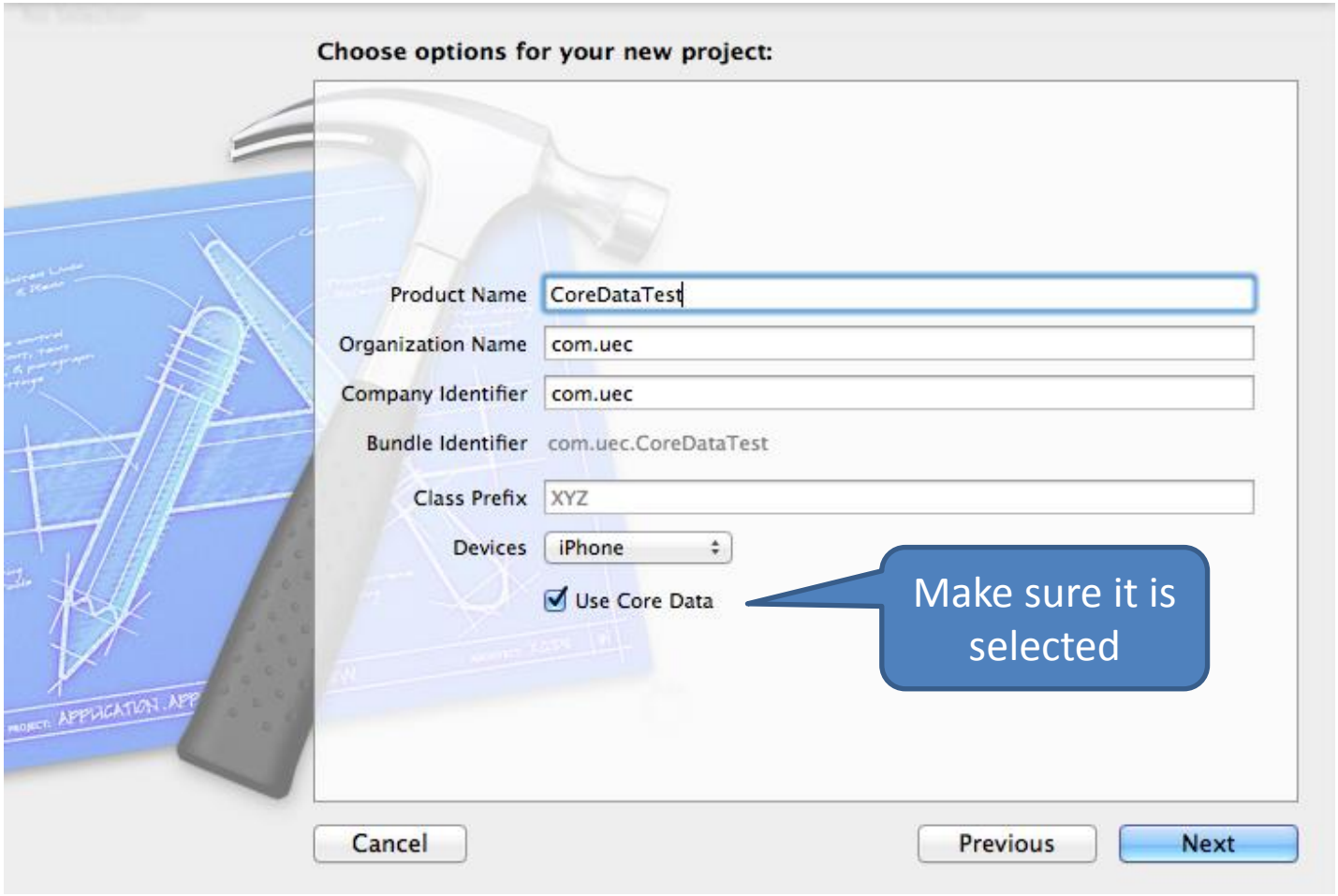
Core Data Stack

- An external persistent store that contains saved records.
- A persistent object store that maps between records in the store and objects in your application.
- A persistent store coordinator that aggregates all the stores.
- A managed object model that describes the entities in the stores.
- A managed object context that provides a scratch pad for managed objects.

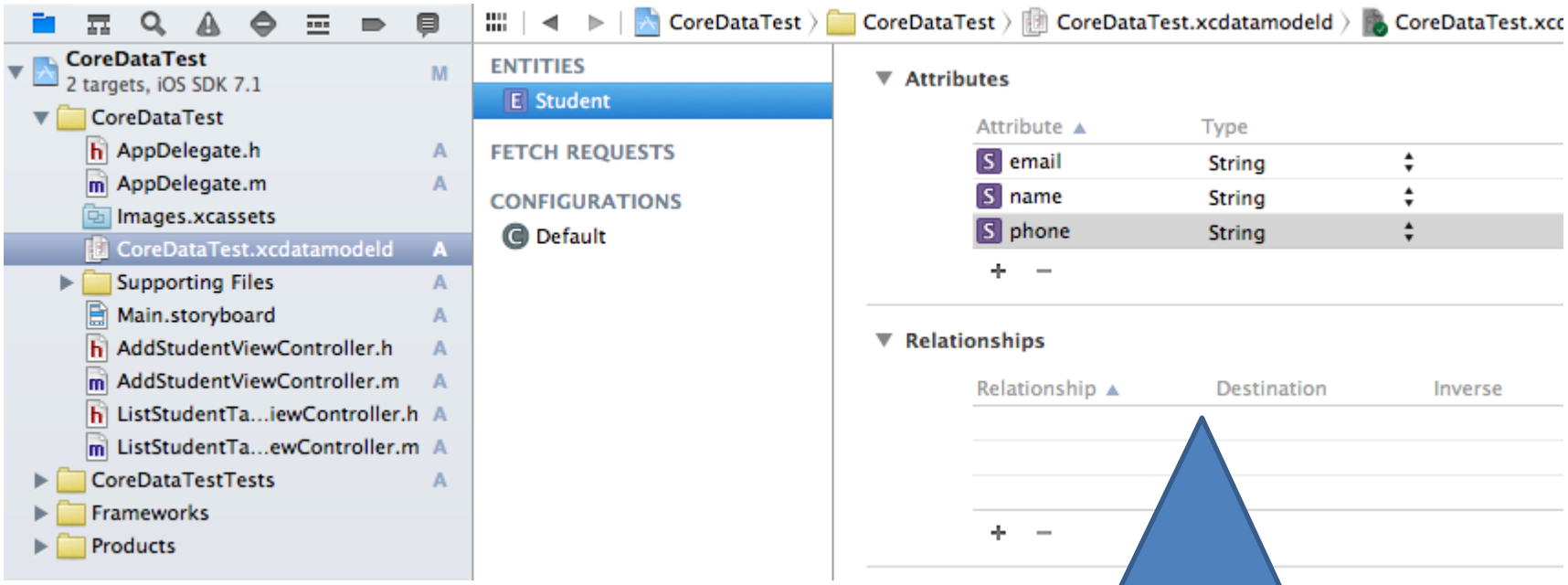
Create a CoreData Example



With Core Data Option

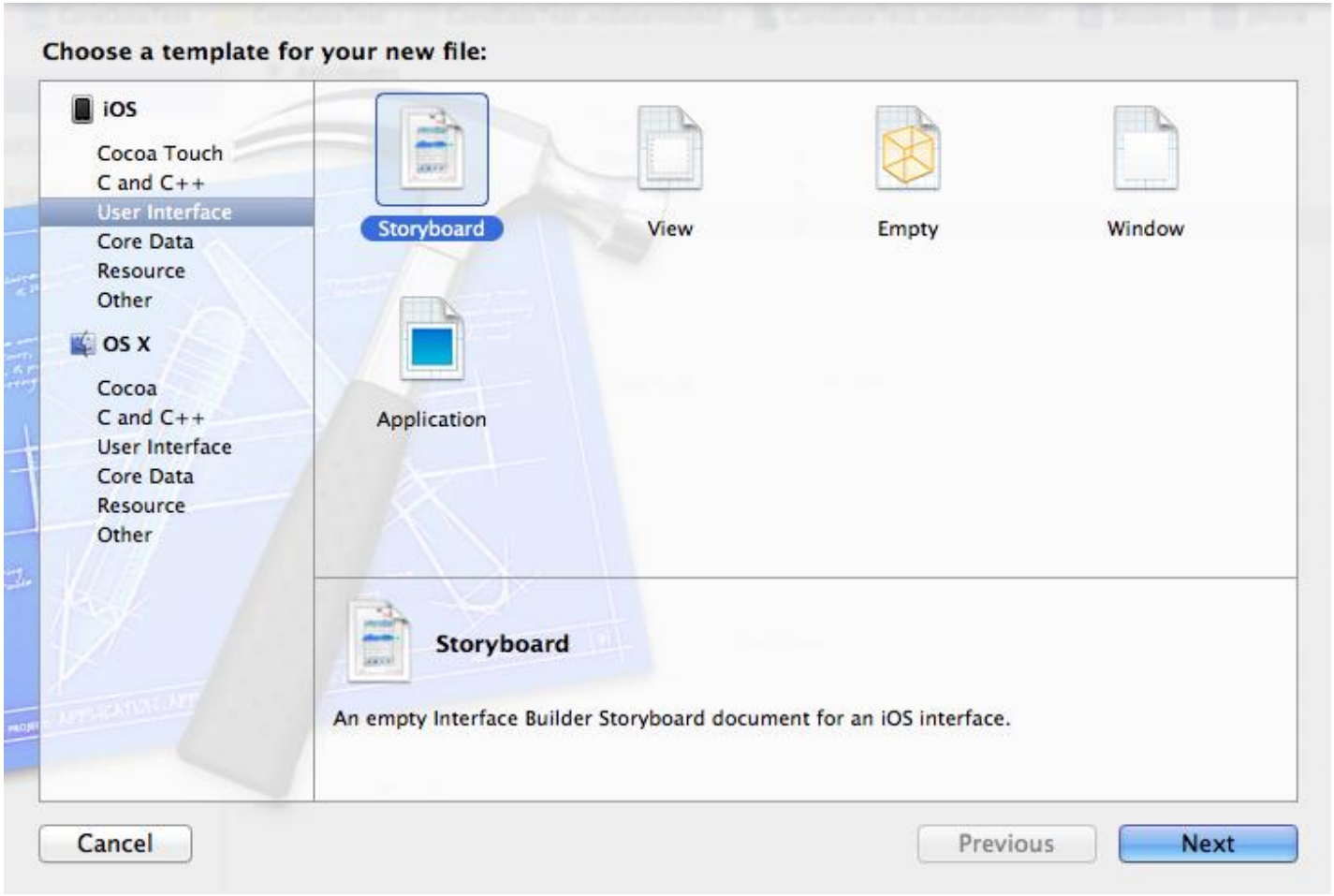


Create an Entity called Student

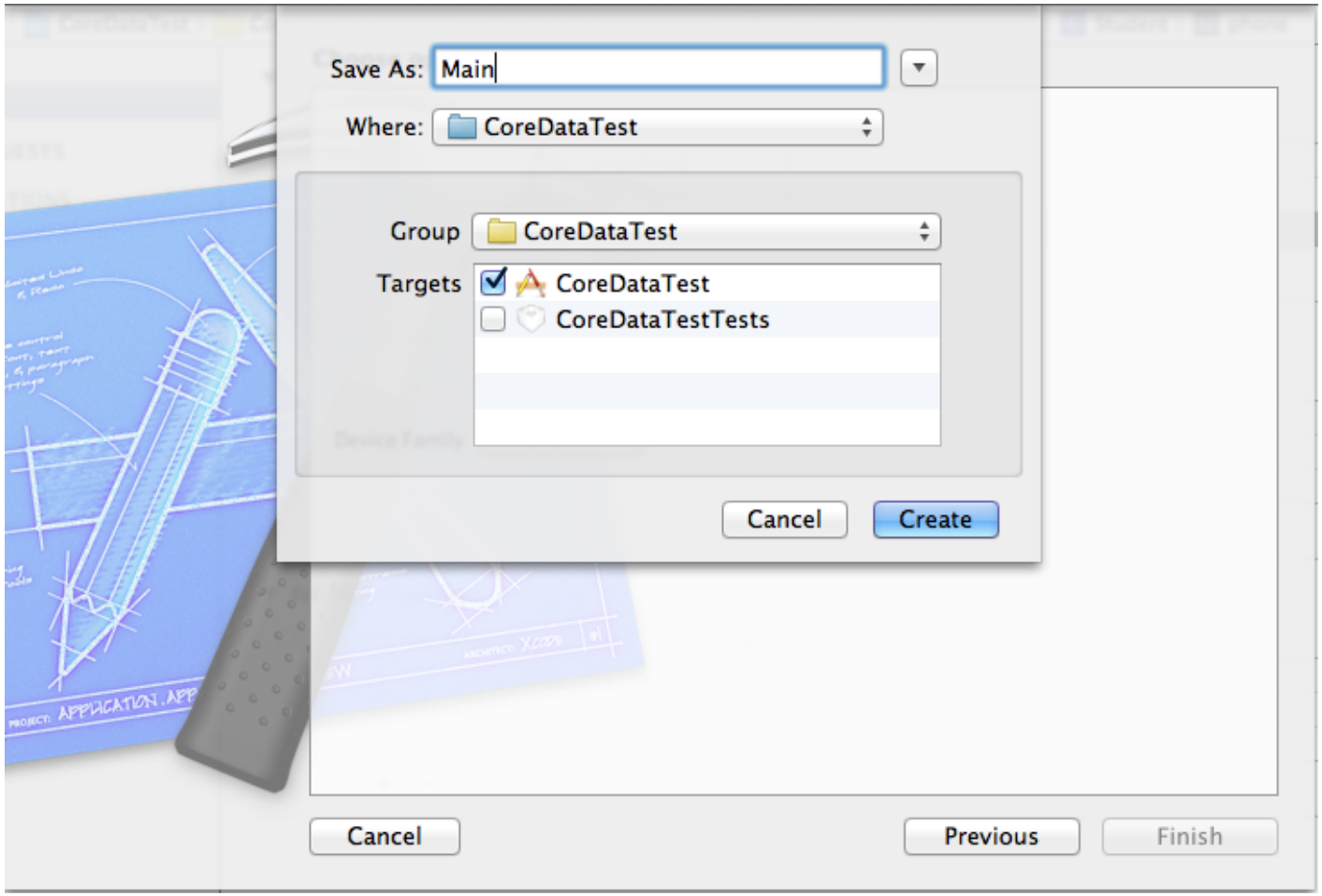


Add 3 properties to the Entity

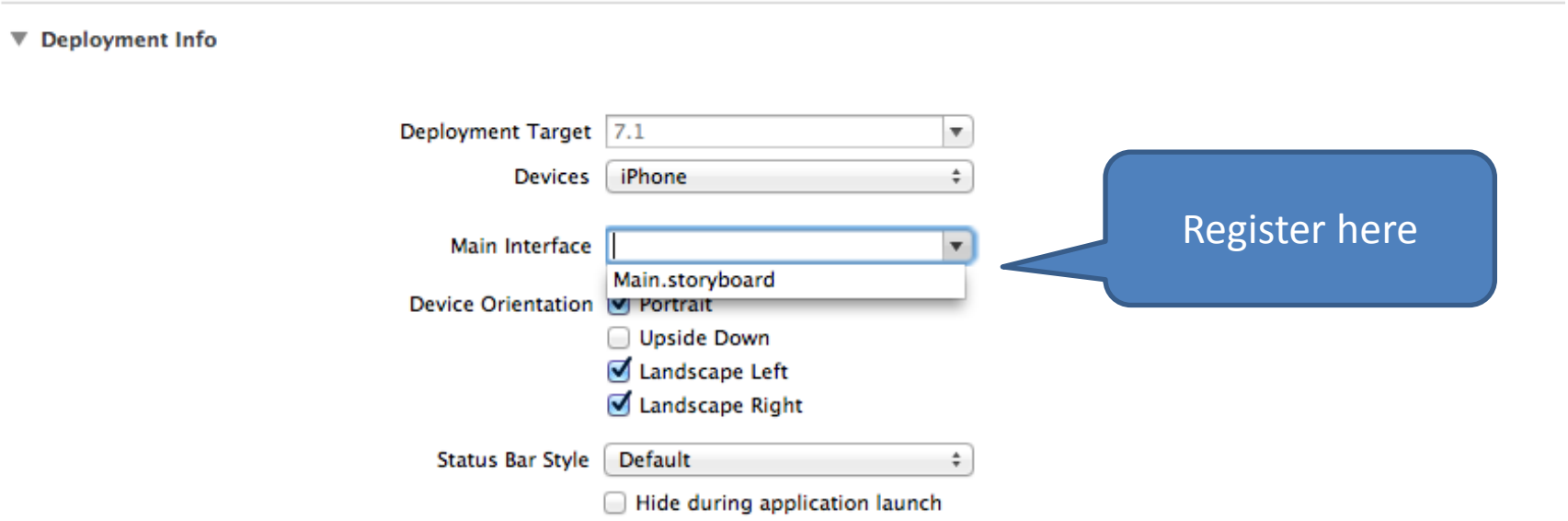
Create a Blank Storyboard



Save the StoryBoard as Main



Make the Main Interface as the new created Storyboard file



Check the AppDelegate.h file

```
//
//  AppDelegate.h
//  CoreDataTest
//
//  Created by Dannis Mok on 29/9/14.
//  Copyright (c) 2014 com.uec. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface AppDelegate : UIResponder <UIApplicationDelegate>

@property (strong, nonatomic) UIWindow *window;

@property (readonly, strong, nonatomic) NSManagedObjectContext *managedObjectContext;
@property (readonly, strong, nonatomic) NSManagedObjectModel *managedObjectModel;
@property (readonly, strong, nonatomic) NSPersistentStoreCoordinator *persistentStoreCoordinator;

- (void)saveContext;
- (NSURL *)applicationDocumentsDirectory;

@end
```

Modify the AppDelegate.m file

```
//
//  AppDelegate.m
//  CoreDataTest
//
//  Created by Dannis Mok on 29/9/14.
//  Copyright (c) 2014 com.uec. All rights reserved.
//

#import "AppDelegate.h"

@implementation AppDelegate

@synthesize managedObjectContext = _managedObjectContext;
@synthesize managedObjectModel = _managedObjectModel;
@synthesize persistentStoreCoordinator = _persistentStoreCoordinator;

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)
    launchOptions
{

    return YES;
}
```

Modify this method

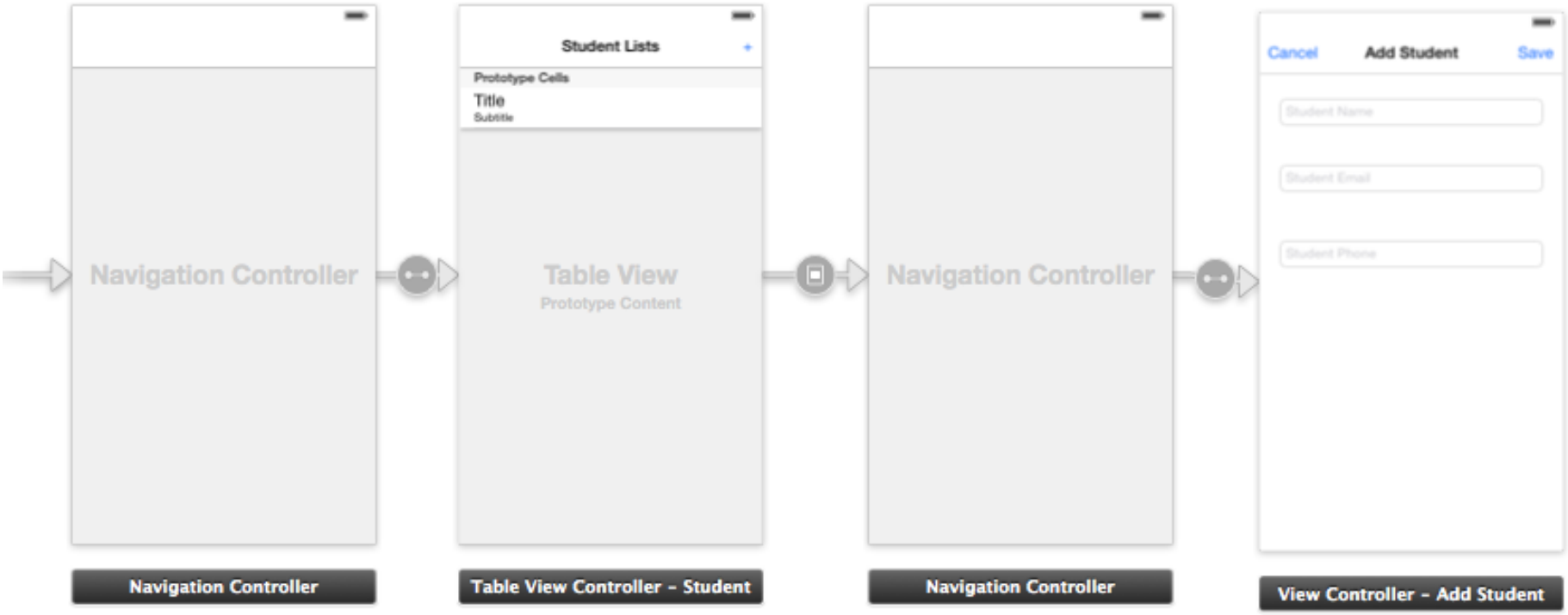
Core Data methods are added in AppDelegate.m file automatically

```
#pragma mark - Core Data stack

// Returns the managed object context for the application.
// If the context doesn't already exist, it is created and bound to the persistent store
    coordinator for the application.
- (NSManagedObjectContext *)managedObjectContext
{
    if (_managedObjectContext != nil) {
        return _managedObjectContext;
    }

    NSPersistentStoreCoordinator *coordinator = [self persistentStoreCoordinator];
    if (coordinator != nil) {
        _managedObjectContext = [[NSManagedObjectContext alloc] init];
        [_managedObjectContext setPersistentStoreCoordinator:coordinator];
    }
    return _managedObjectContext;
}

// Returns the managed object model for the application.
// If the model doesn't already exist, it is created from the application's model.
- (NSManagedObjectModel *)managedObjectModel
{
    if (_managedObjectModel != nil) {
        return _managedObjectModel;
    }
    NSURL *modelURL = [[NSBundle mainBundle] URLForResource:@"CoreDataTest" withExtension:@"momd"
        ];
    _managedObjectModel = [[NSManagedObjectModel alloc] initWithContentsOfURL:modelURL];
    return _managedObjectModel;
}
```

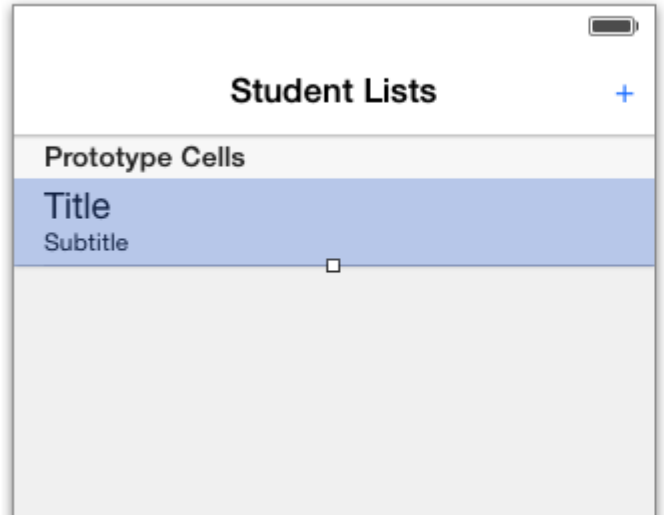
Core Data methods are added in AppDelegate.m file automatically

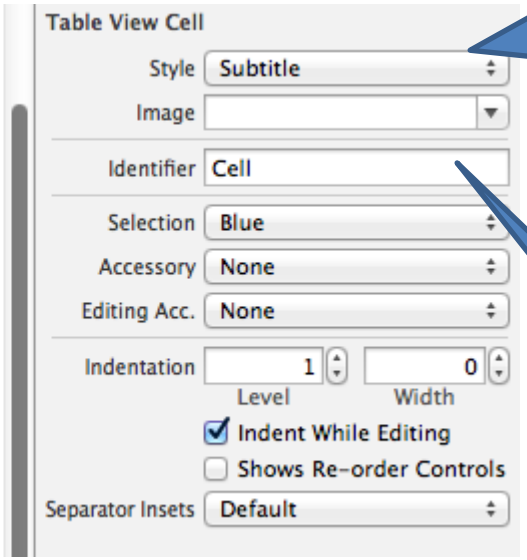
```
// Returns the persistent store coordinator for the application.
// If the coordinator doesn't already exist, it is created and the application's store added to
    it.
- (NSPersistentStoreCoordinator *)persistentStoreCoordinator
{
    if (_persistentStoreCoordinator != nil) {
        return _persistentStoreCoordinator;
    }

    NSURL *storeURL = [[self applicationDocumentsDirectory]
        URLByAppendingPathComponent:@"CoreDataTest.sqlite"];

    NSError *error = nil;
    _persistentStoreCoordinator = [[NSPersistentStoreCoordinator alloc]
        initWithManagedObjectModel:[self managedObjectModel]];
    if (![_persistentStoreCoordinator addPersistentStoreWithType:NSSQLiteStoreType configuration:
        nil URL:storeURL options:nil error:&error]) {
        /*
         Replace this implementation with code to handle the error appropriately.
```

Create the interfaces as below



Update the Table View Cell settings

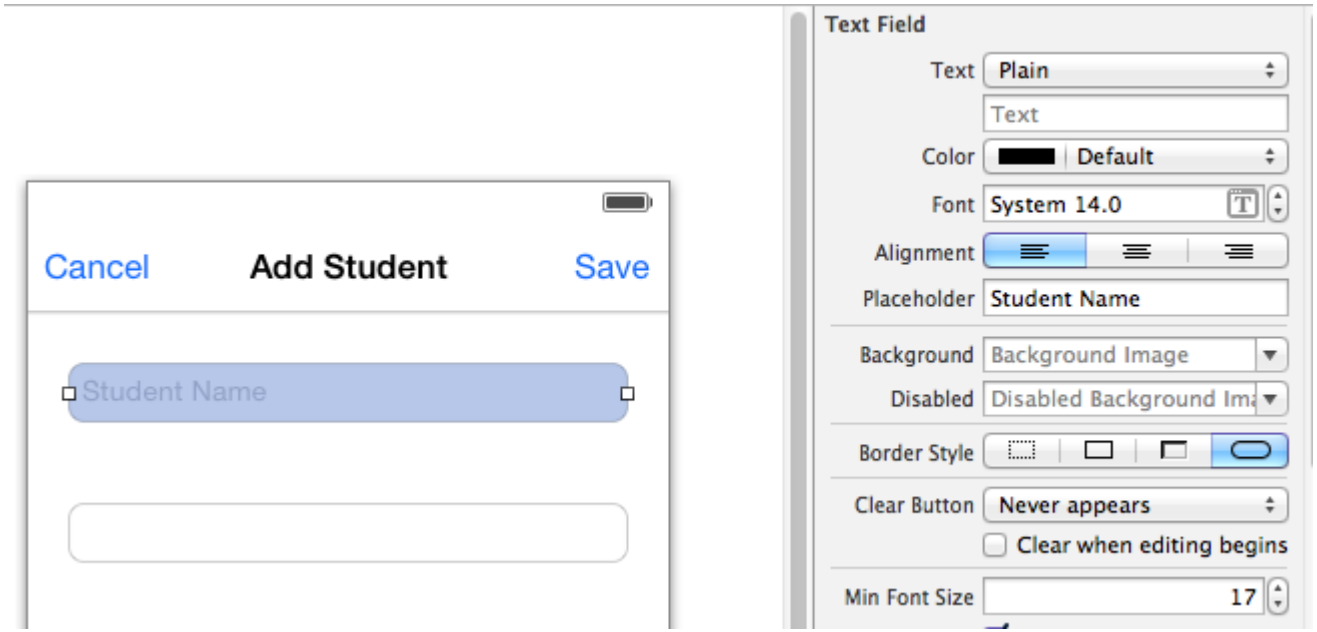




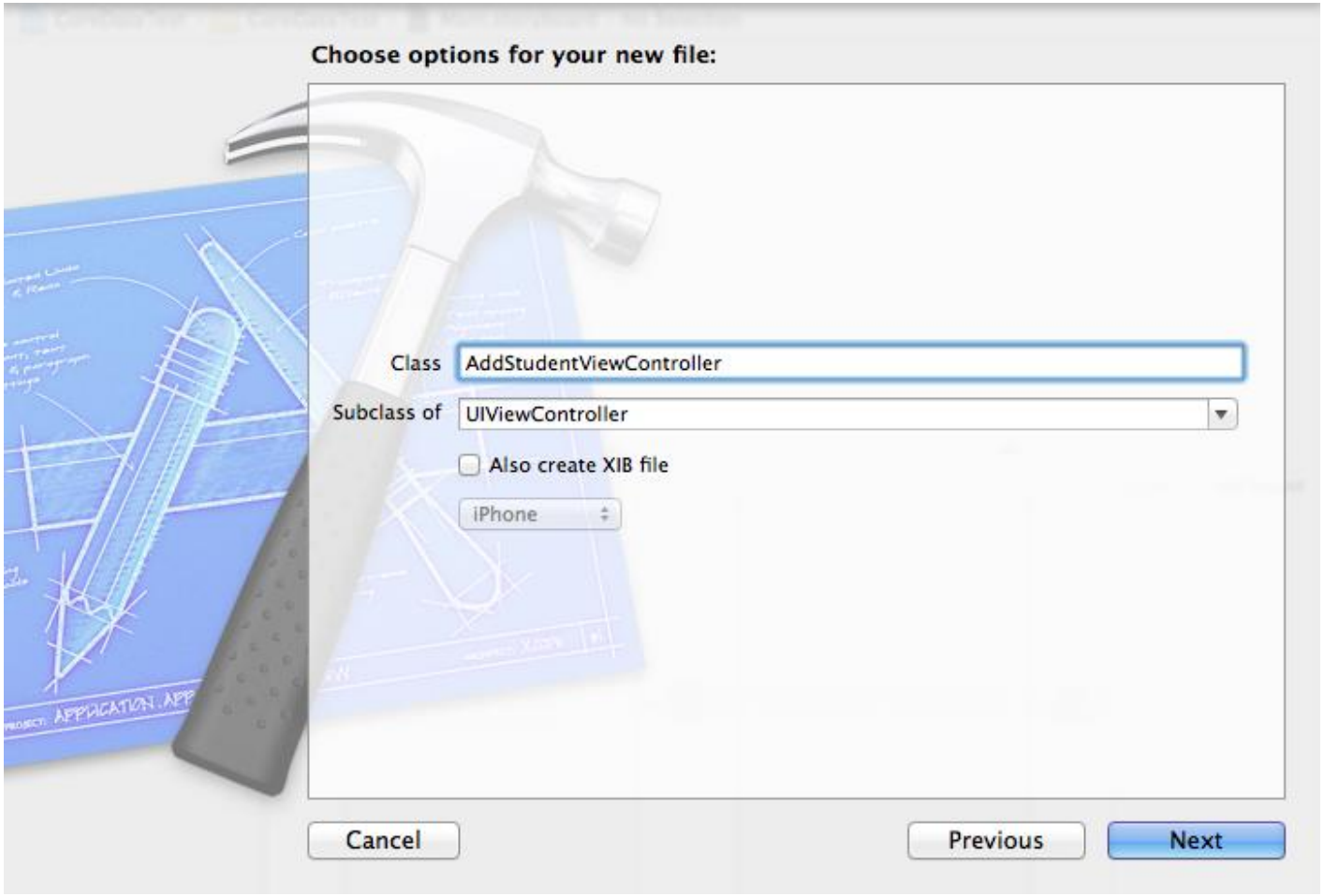
Change the display style

Add the Cell Identifier

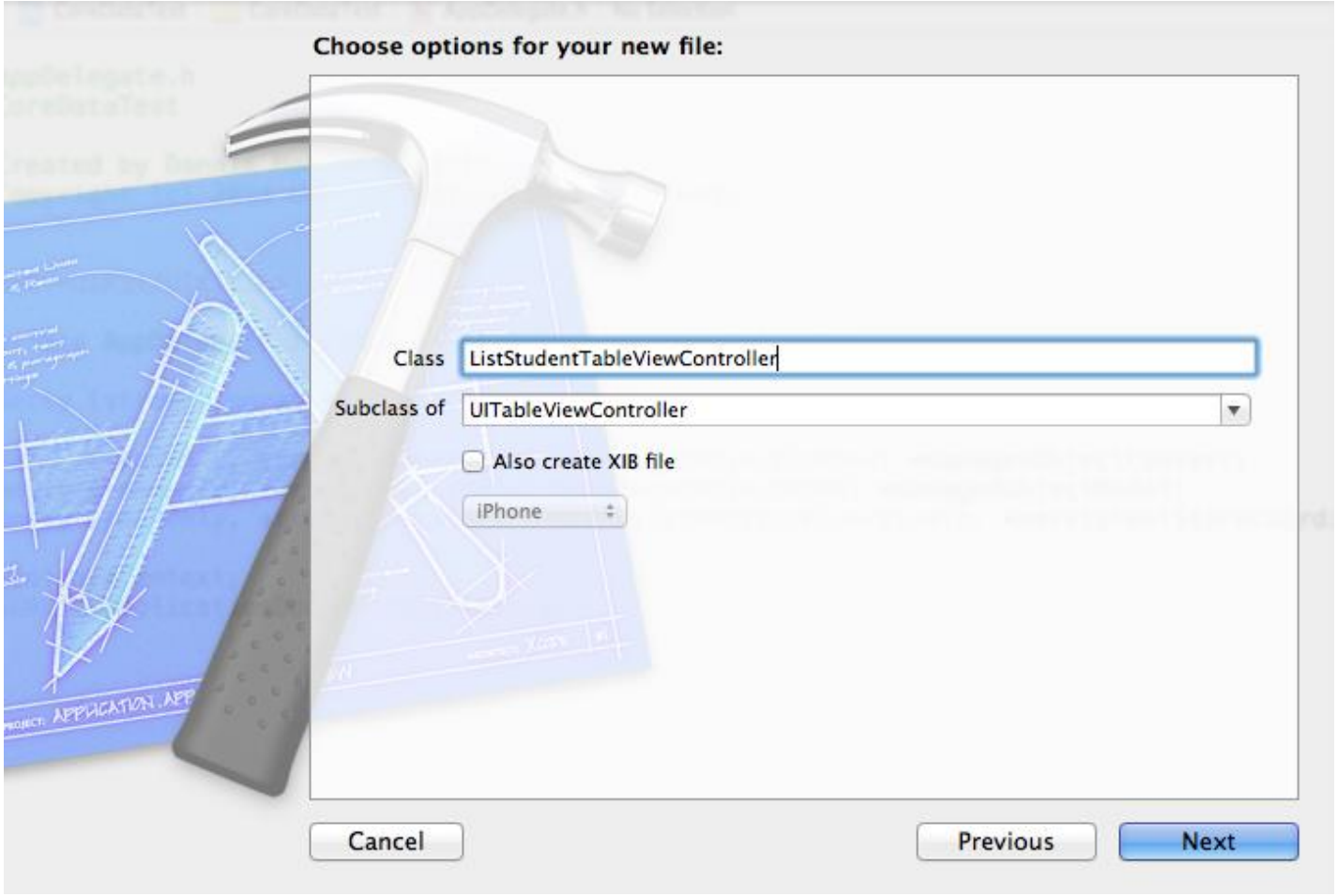
Add TextField and Buttons to the Add Student View



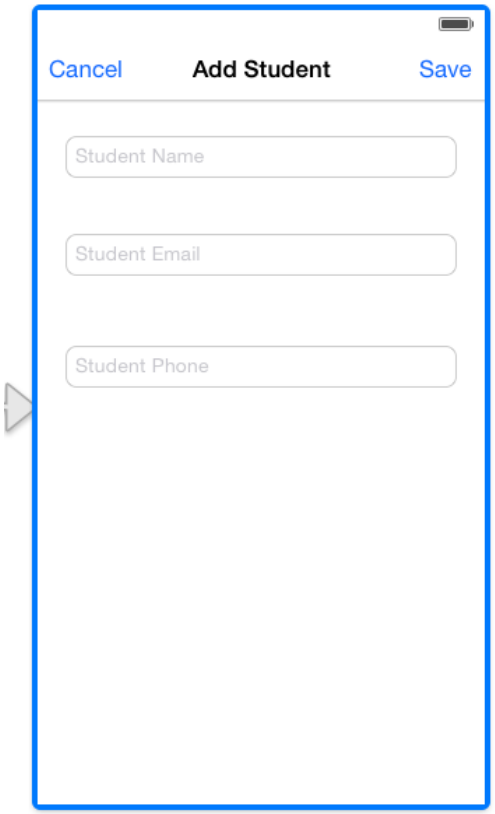
Add UIViewController file for the Add Student View Controller

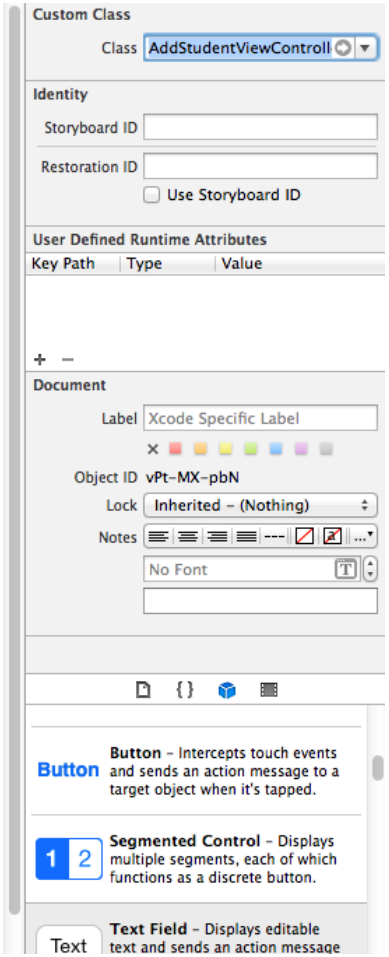


Add UITableViewController file for the TableViewController View

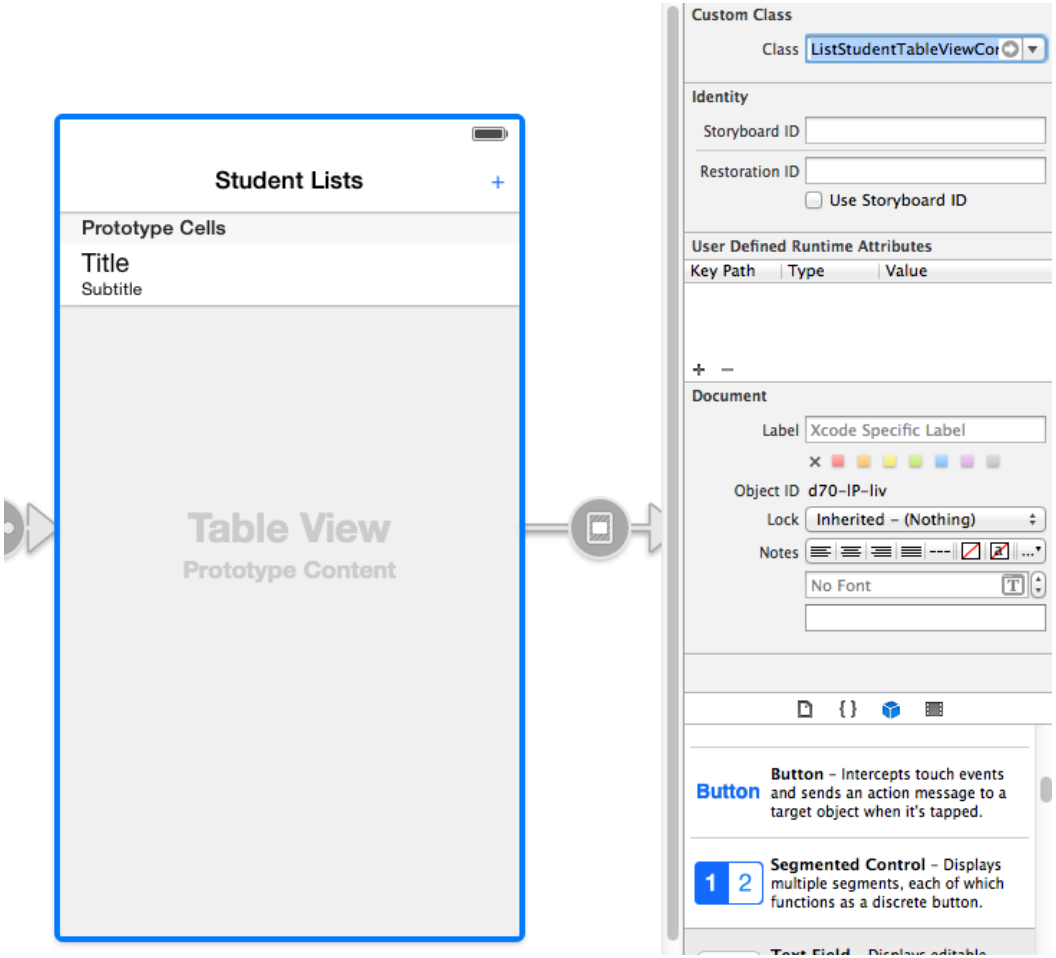


Link the view to the controller file

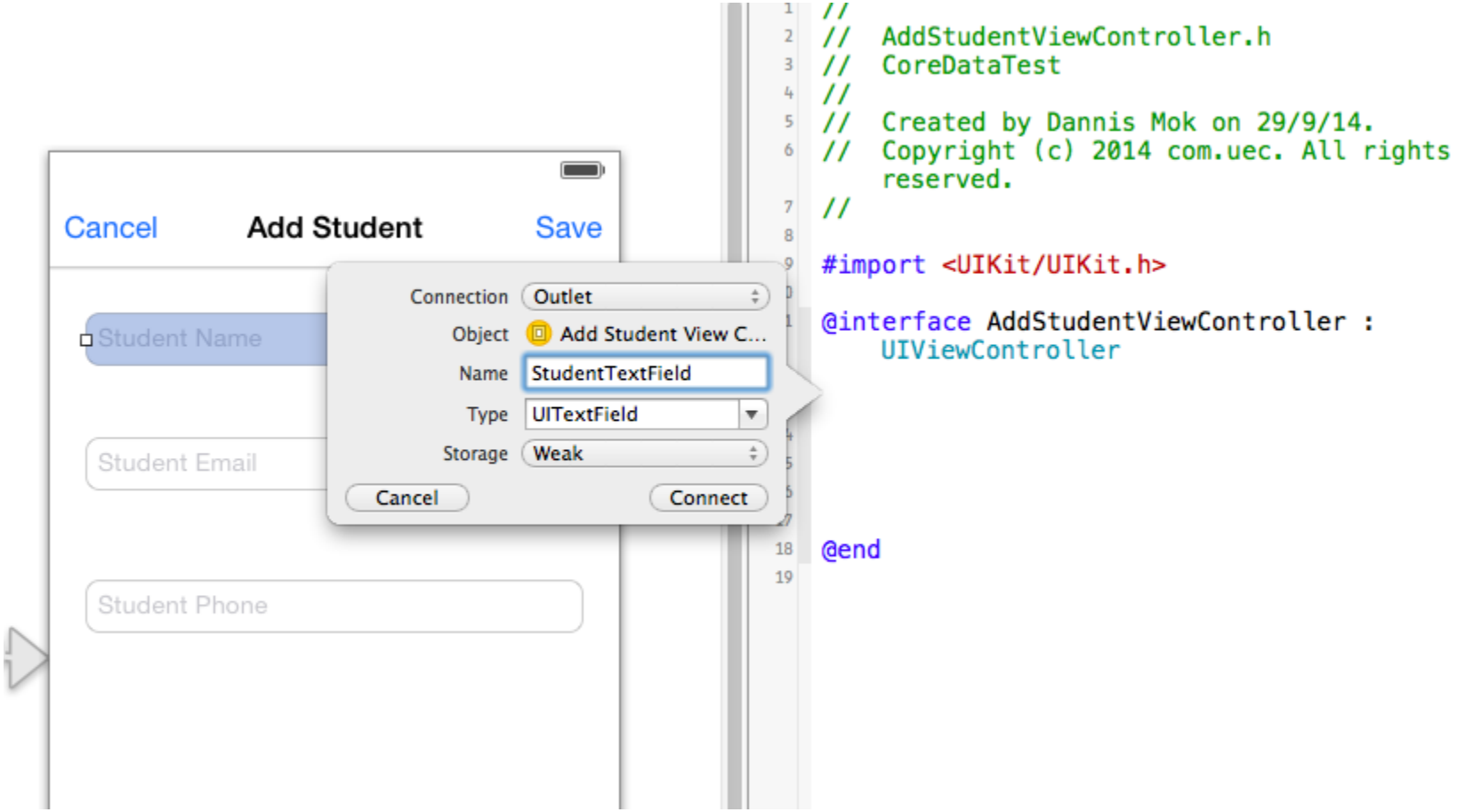




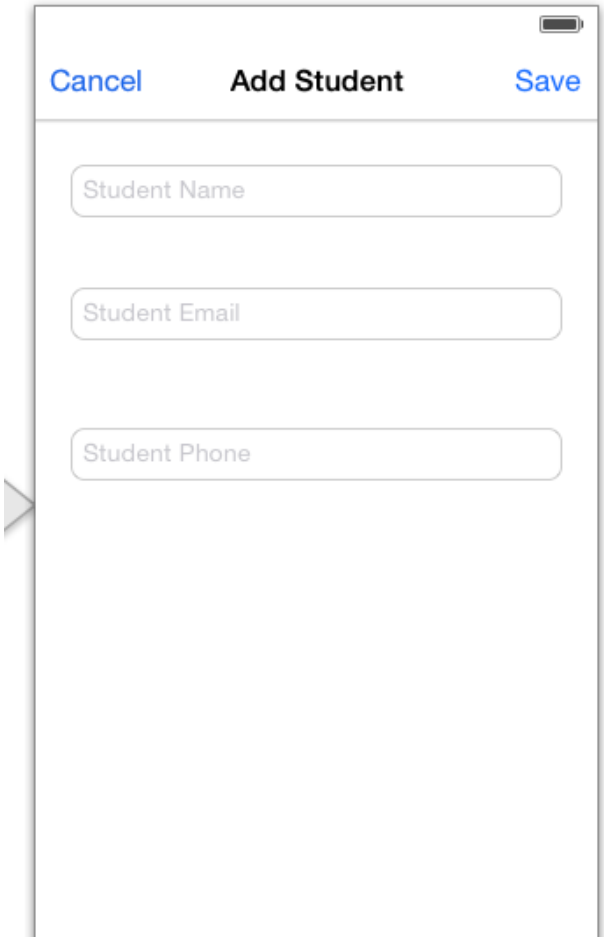
Link the view to the controller file



Add Outlets for the field



Outlets and Actions are added




```
//
//  AddStudentViewController.h
//  CoreDataTest
//
//  Created by Dannis Mok on 29/9/14.
//  Copyright (c) 2014 com.uec. All rights
    reserved.
//

#import <UIKit/UIKit.h>

@interface AddStudentViewController :
    UIViewController


@property (weak, nonatomic) IBOutlet
    UITextField *nameTextField;


@property (weak, nonatomic) IBOutlet
    UITextField *emailTextField;


@property (weak, nonatomic) IBOutlet
    UITextField *phoneTextField;



- (IBAction)save:(id)sender;


- (IBAction)cancel:(id)sender;

@end
```

Add the dismissViewController methods to go back

```
- (IBAction)save:(id)sender {

    [self dismissViewControllerAnimated:YES completion:nil];

}

- (IBAction)cancel:(id)sender {


    [self dismissViewControllerAnimated:YES completion:nil];
}
@end
```

Core Data related coding

```
-(NSManagedObjectContext *) managedObjectContext {
    NSManagedObjectContext *context = nil;
    id delegate = [[UIApplication sharedApplication] delegate];
    if([delegate performSelector:@selector(managedObjectContext)]) {
        context = [delegate managedObjectContext];
    }
    return context;
}
```

Add the code in the AddStudentViewController.m to get back the NSManagedObjectContext

Update the Save method

```
- (IBAction)save:(id)sender {

    NSManagedObjectContext *context = [self managedObjectContext];

    NSManagedObject *newStudent = [NSEntityDescription insertNewObjectForEntityForName:@"Student"
        inManagedObjectContext:context];

    [newStudent setValue:self.nameTextField.text forKey:@"name"];

    [newStudent setValue:self.emailTextField.text forKey:@"email"];

    [newStudent setValue:self.phoneTextField.text forKey:@"phone"];


    NSError *error = nil;

    if(![context save:&error]) {

        NSLog(@"Can't Save ! %@ %@",error,[error localizedDescription]);

    }

    [self dismissViewControllerAnimated:YES completion:nil];

}
```

Create a new NSManagedObject and then save the values

Core Data coding in TableView

```
#import "ListStudentTableViewController.h"
@interface ListStudentTableViewController ()
@end
@implementation ListStudentTableViewController  {
    NSMutableArray *students;
}

-(NSManagedObjectContext *) managedObjectContext {
    NSManagedObjectContext *context = nil;
    id delegate = [[UIApplication sharedApplication] delegate];
    if([delegate performSelector:@selector(managedObjectContext)]) {
        context = [delegate managedObjectContext];
    }
    return context;
}
```

Create an Array

Add the code in the TableViewController.m to get back the NSManagedObjectContext

Add the code when view appear

```
-(void)viewDidAppear:(BOOL)animated
{

    [super viewDidAppear:YES];

    NSManagedObjectContext *managedObjectContext = [self managedObjectContext];

    NSFetchRequest *fetchRequest = [[NSFetchRequest alloc]initWithEntityName:@"Student"];

    students =[[managedObjectContext executeFetchRequest:fetchRequest error:nil]mutableCopy];

    [self.tableView reloadData];

}
```

Add above method to update the array and also refresh the view

Modify the TableView codes

```
#pragma mark - Table view data source

- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
#warning Potentially incomplete method implementation.
    // Return the number of sections.
    return 1;
}

- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{
#warning Incomplete method implementation.
    // Return the number of rows in the section.
    return [students count];
}
```

Update the number of sections to 1

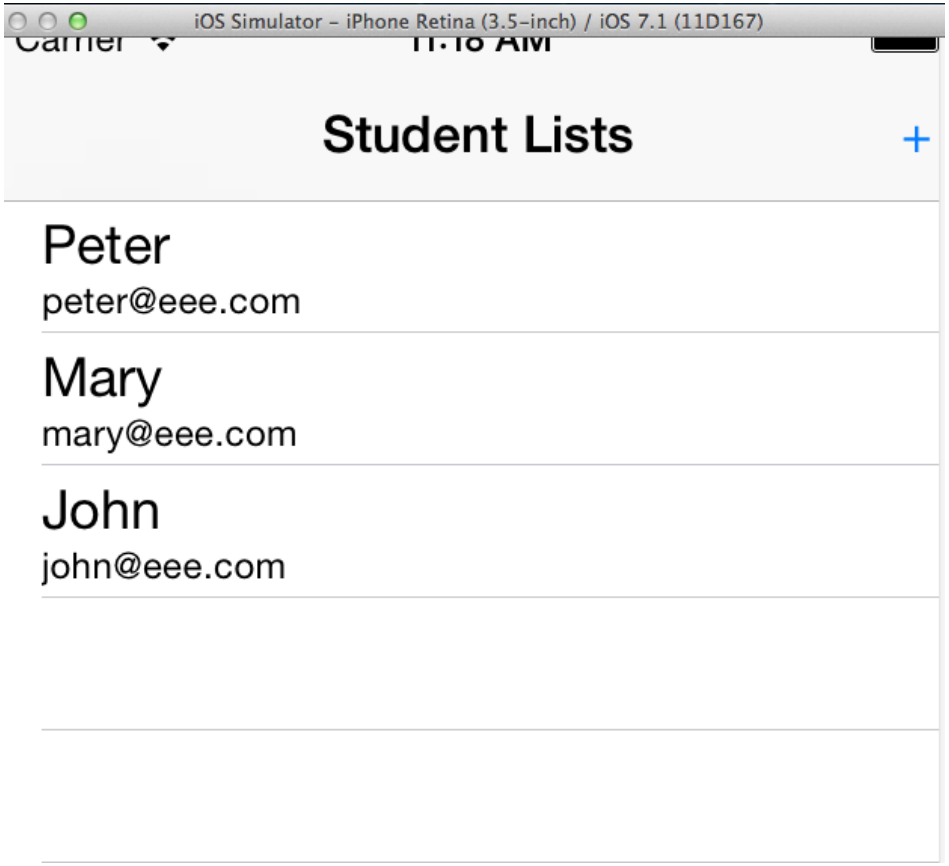
Update the number of rows to array size

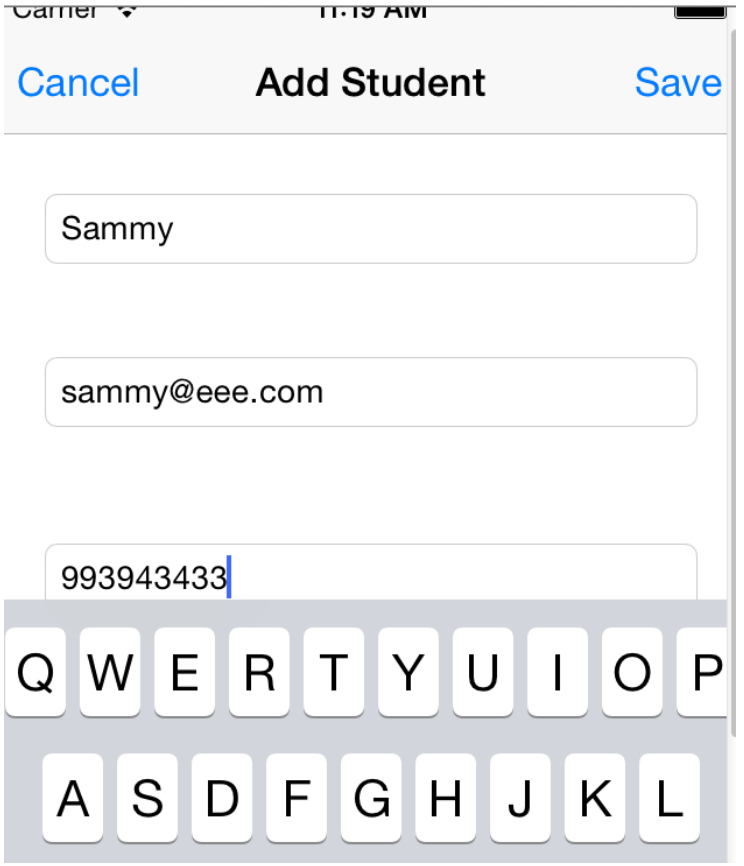
Modify the TableView codes

```
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath
    *)indexPath
{
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:@"Cell" forIndexPath:
        indexPath];

    NSManagedObject *student = [students objectAtIndex:indexPath.row];

    cell.textLabel.text = [student valueForKey:@"name"];

    cell.detailTextLabel.text = [student valueForKey:@"email"];

    return cell;
}
```

Get the data from the array and then create a new cell to display it

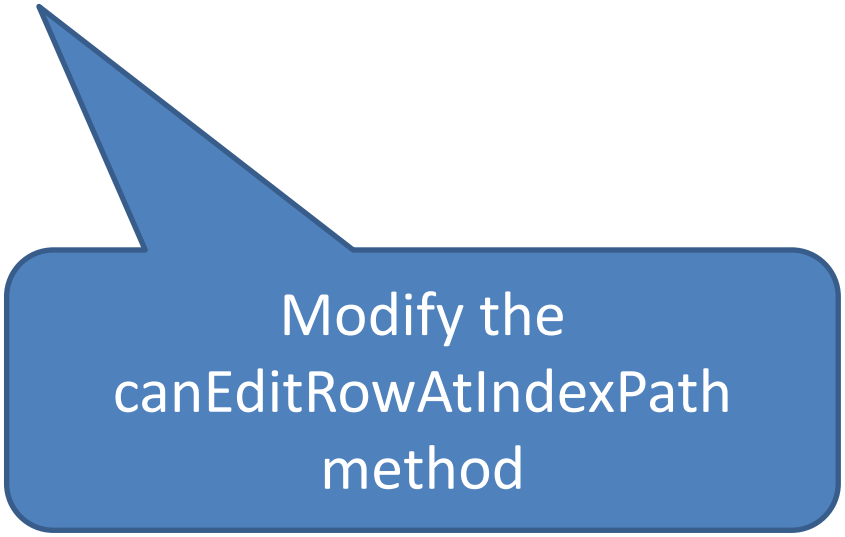
Screenshots of Adding and Listing Students





Modify the code to support Deletion

```
// Override to support conditional editing of the table view.
- (BOOL)tableView:(UITableView *)tableView canEditRowAtIndexPath:(NSIndexPath *)indexPath
{
    // Return NO if you do not want the specified item to be editable.
    return YES;
}
```

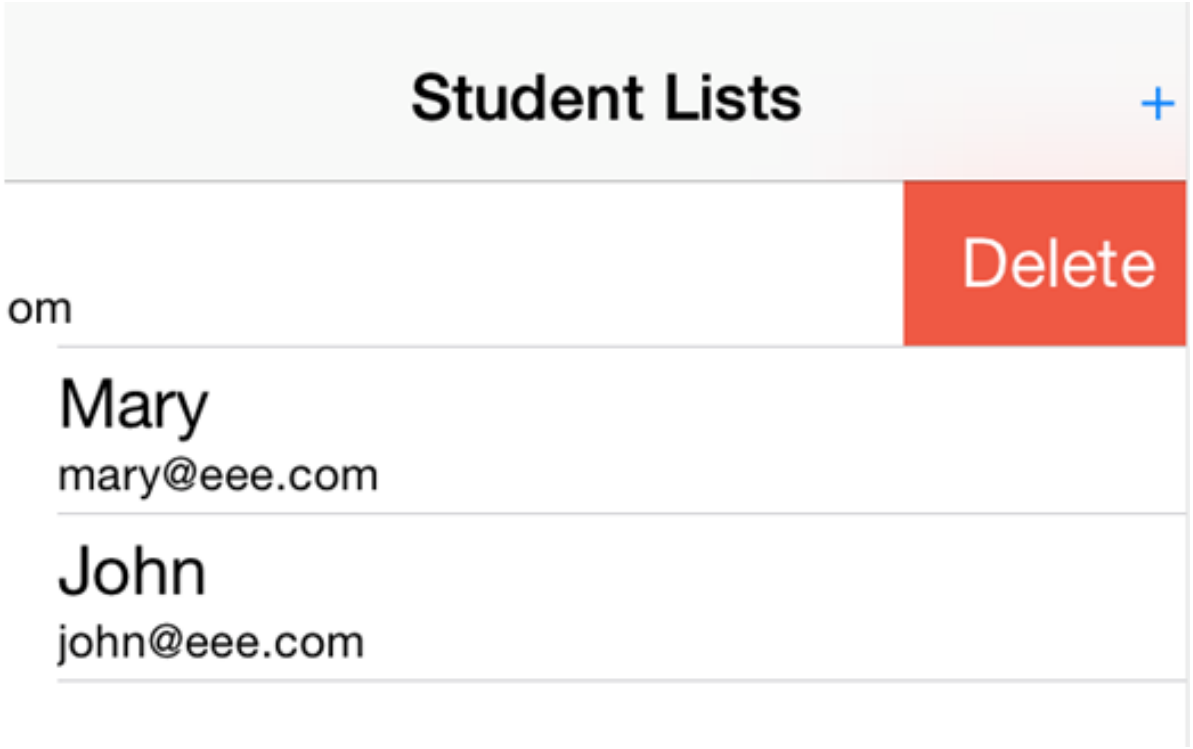


Modify the code to support Deletion

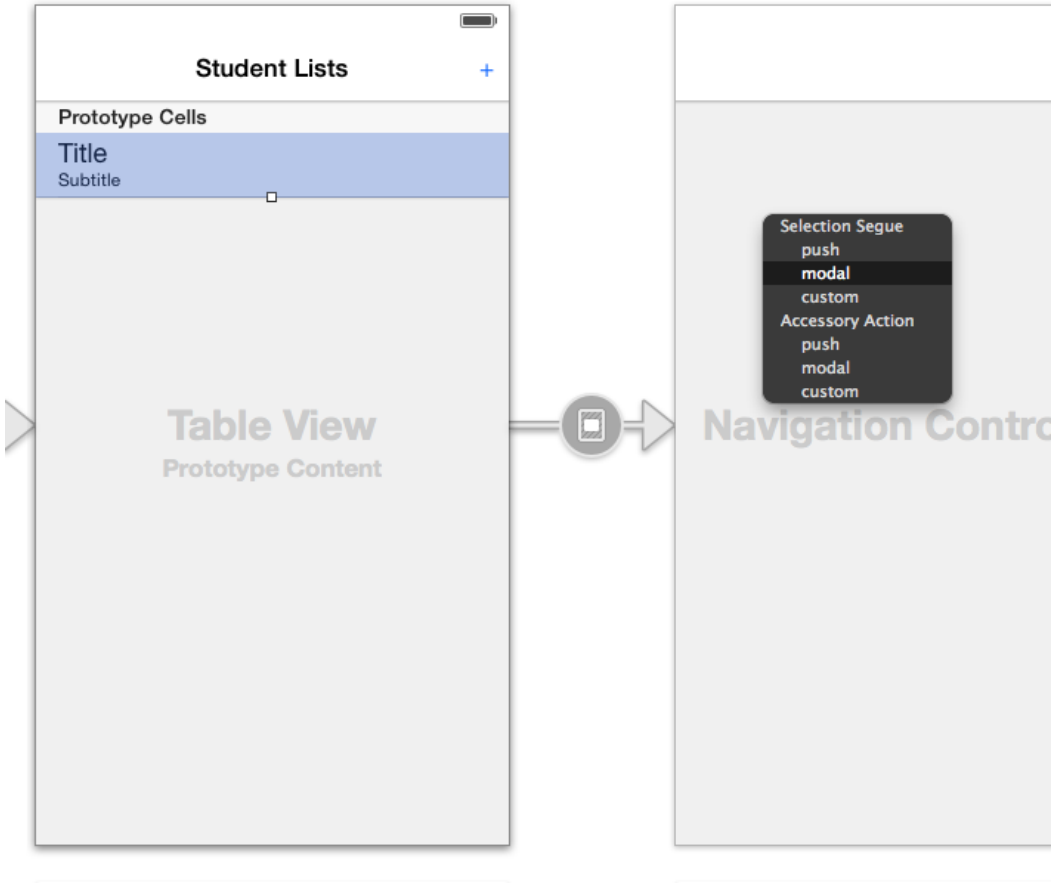
```
// Override to support editing the table view.
- (void)tableView:(UITableView *)tableView commitEditingStyle:
    (UITableViewCellEditingStyle)editingStyle forRowAtIndexPath:(NSIndexPath *)indexPath
{

    NSManagedObjectContext *context = [self managedObjectContext];

    if(editingStyle == UITableViewCellEditingStyleDelete) {

        [context deleteObject:[students objectAtIndex:indexPath.row]];

        NSError *error = nil;

        if(![context save:&error]) {

            NSLog(@"Can't Delete ! %@ %@",error, [error localizedDescription]);

        }

        [students removeObjectAtIndex:indexPath.row];

        [self.tableView deleteRowsAtIndexPaths:[NSArray arrayWithObject:indexPath]
            withRowAnimation:UITableViewRowAnimationFade];

    }

}
```

Modify the commitEditingStyle method

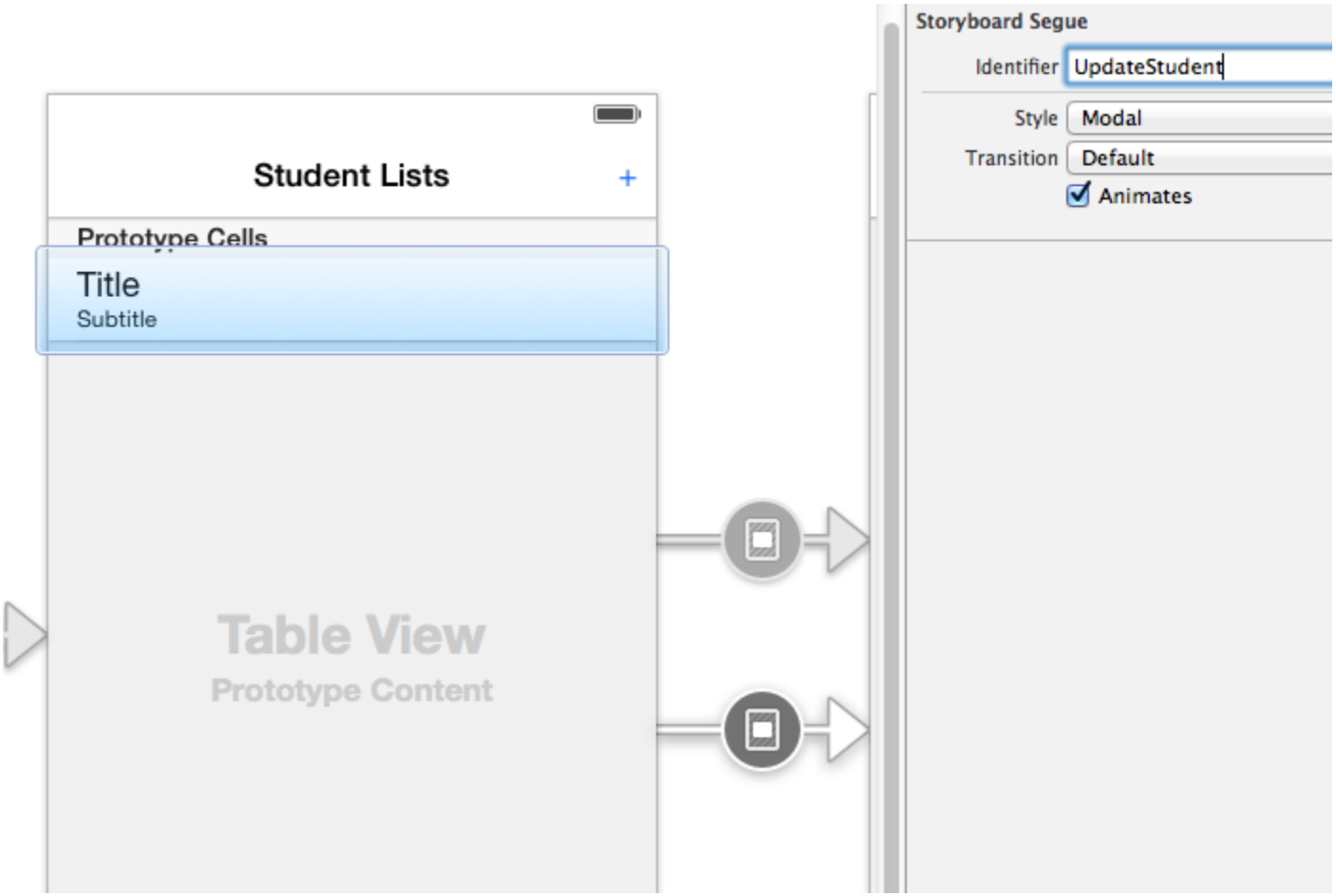
Screenshot of Deleting Student



Add another Push Segue from each cell



Name the Segue



Modify the AddStudentViewController

```
//
//  AddStudentViewController.h
//  CoreDataTest
//
//  Created by Dannis Mok on 29/9/14.
//  Copyright (c) 2014 com.uec. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface AddStudentViewController : UIViewController

@property (weak, nonatomic) IBOutlet UITextField *nameTextField;

@property (weak, nonatomic) IBOutlet UITextField *emailTextField;

@property (weak, nonatomic) IBOutlet UITextField *phoneTextField;

@property (strong) NSManagedObject *student;

- (IBAction)save:(id)sender;

- (IBAction)cancel:(id)sender;

@end
```

Add an object to store information sent from TableViewController

Modify the code in TableView

```
//
//  ListStudentTableViewController.m
//  CoreDataTest
//
//  Created by Dannis Mok on 29/9/14.
//  Copyright (c) 2014 com.uec. All rights reserved.
//

#import "ListStudentTableViewController.h"

#import "AddStudentViewController.h"
```

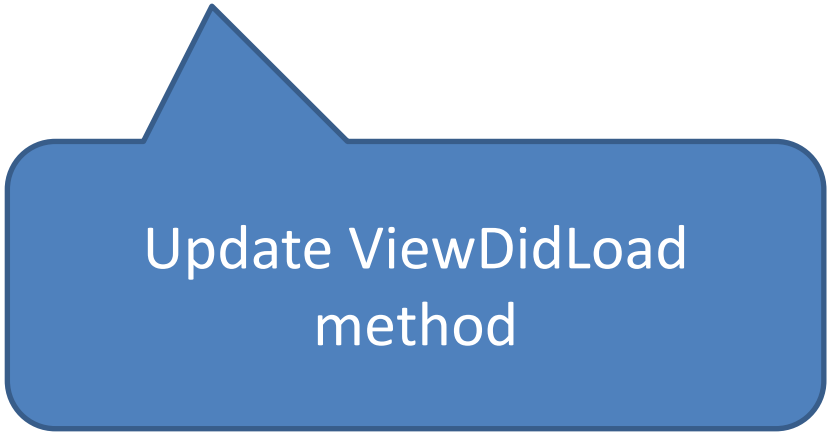
Add destination view controller header file

Send the object to destination view controller

```
#pragma mark - Navigation

// In a storyboard-based application, you will often want to do a little preparation before
    navigation
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender
{


    if([[segue identifier] isEqualToString:@"UpdateStudent"]) {

        NSManagedObject *selectedStudent = [students objectAtIndex:[[self.tableView
            indexPathForSelectedRow]row]];

        UINavigationController *destViewController = segue.destinationViewController;

        AddStudentViewController *studentViewController = (AddStudentViewController *)
            destViewController.topViewController;


        studentViewController.student = selectedStudent;


    }


}
```

Update the AddStudentViewController

```
- (void)viewDidLoad
{
    [super viewDidLoad];

    if(self.student) {

        [self.nameTextField setText:[self.student valueForKey:@"name"]];

        [self.emailTextField setText:[self.student valueForKey:@"email"]];

        [self.phoneTextField setText:[self.student valueForKey:@"phone"]];

    }
}
```

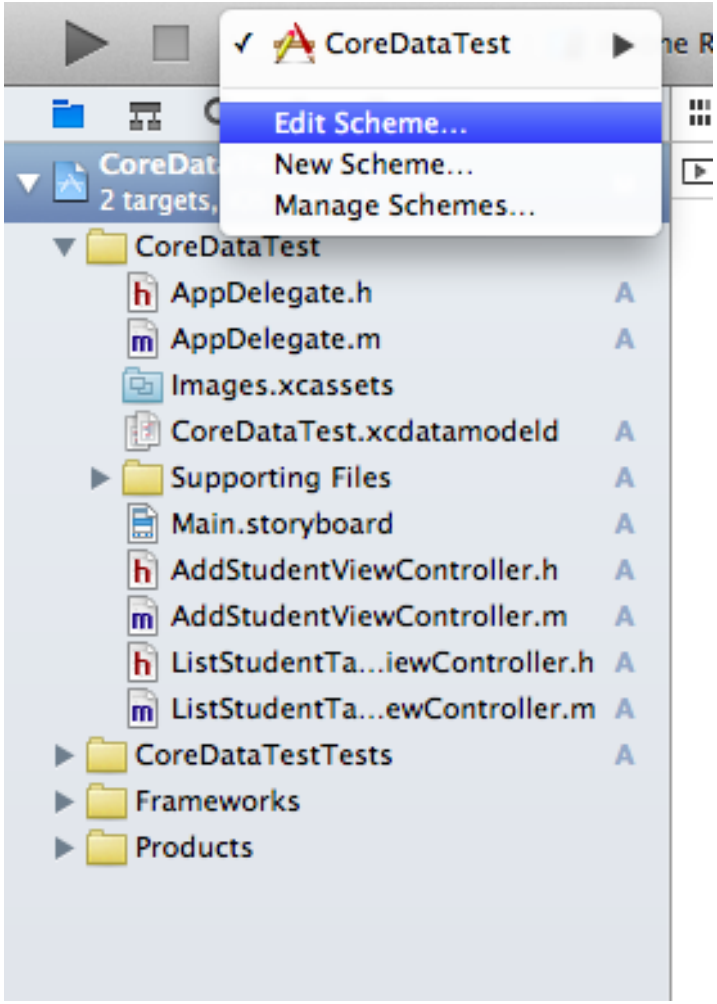


Update the Save Method

```
- (IBAction)save:(id)sender {

    NSManagedObjectContext *context = [self managedObjectContext];

    if(self.student) {

        [self.student setValue:self.nameTextField.text forKey:@"name"];

        [self.student setValue:self.emailTextField.text forKey:@"email"];

        [self.student setValue:self.phoneTextField.text forKey:@"phone"];

    } else {

        NSManagedObject *newStudent = [NSEntityDescription
            insertNewObjectForEntityForName:@"Student" inManagedObjectContext:context];

       [newStudent setValue:self.nameTextField.text forKey:@"name"];

       [newStudent setValue:self.emailTextField.text forKey:@"email"];

       [newStudent setValue:self.phoneTextField.text forKey:@"phone"];

    }

    NSError *error = nil;

    if(![context save:&error]) {

        NSLog(@"Can't Save ! %@ %@",error,[error localizedDescription]);

    }

    [self dismissViewControllerAnimated:YES completion:nil];

}
```

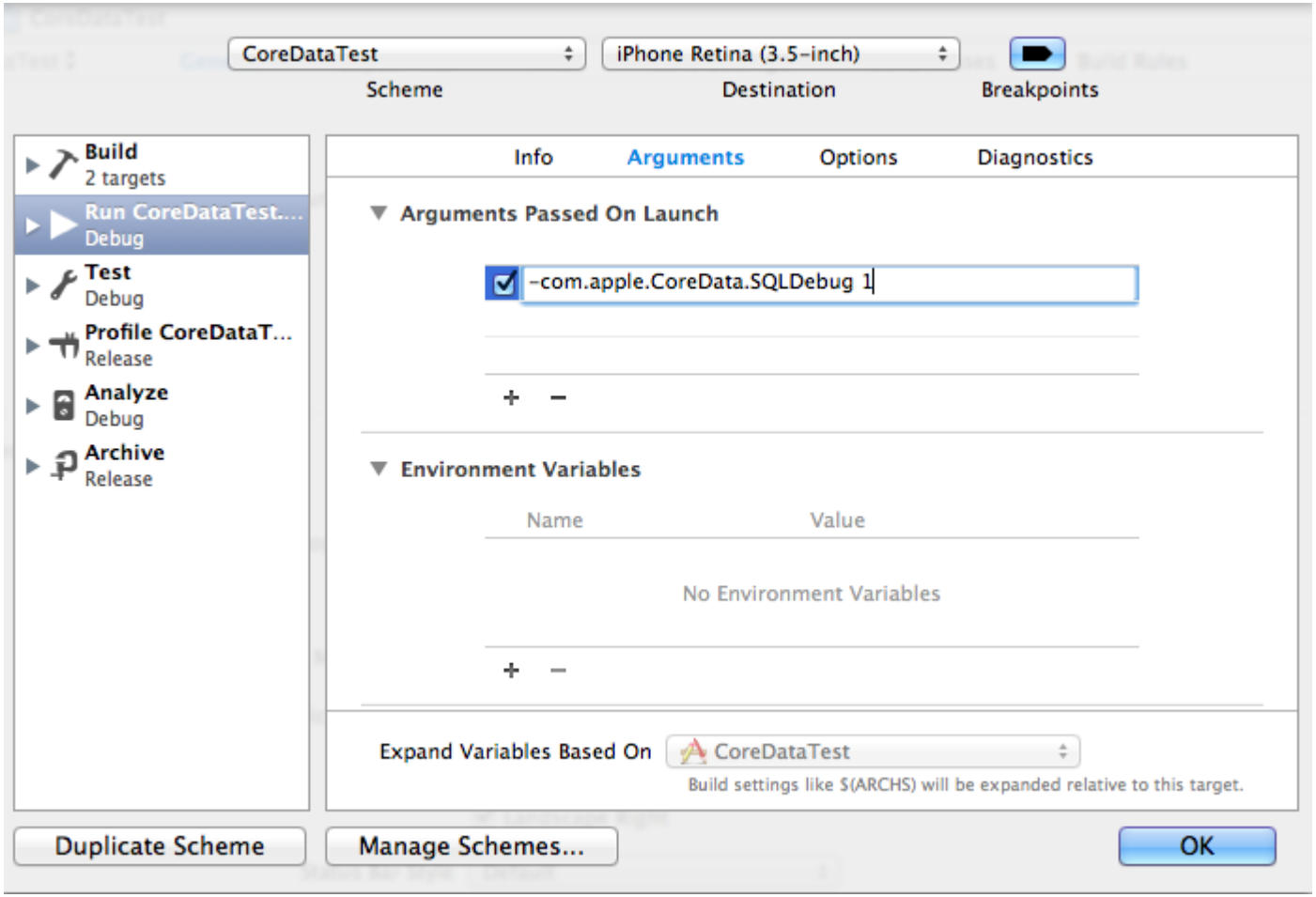
Add these to support Update

Show the SQL commands



Modify the scheme

Add the code as follows



Run the app and see the SQL statements

```
2014-09-29 11:41:00.979 CoreDataTest[3975:90b] CoreData: sql: BEGIN EXCLUSIVE
2014-09-29 11:41:00.980 CoreDataTest[3975:90b] CoreData: sql: SELECT Z_MAX FROM
Z_PRIMARYKEY WHERE Z_ENT = ?
2014-09-29 11:41:00.982 CoreDataTest[3975:90b] CoreData: sql: UPDATE Z_PRIMARYKEY SET
Z_MAX = ? WHERE Z_ENT = ? AND Z_MAX = ?
2014-09-29 11:41:00.983 CoreDataTest[3975:90b] CoreData: sql: COMMIT
2014-09-29 11:41:00.983 CoreDataTest[3975:90b] CoreData: sql: BEGIN EXCLUSIVE
2014-09-29 11:41:00.984 CoreDataTest[3975:90b] CoreData: sql: INSERT INTO
ZSTUDENT(Z_PK, Z_ENT, Z_OPT, ZEMAIL, ZNAME, ZPHONE) VALUES(?, ?, ?, ?, ?, ?)
2014-09-29 11:41:00.985 CoreDataTest[3975:90b] CoreData: sql: COMMIT
2014-09-29 11:41:00.986 CoreDataTest[3975:90b] CoreData: sql: pragma page_count
2014-09-29 11:41:00.987 CoreDataTest[3975:90b] CoreData: annotation: sql execution
time: 0.0006s
2014-09-29 11:41:00.987 CoreDataTest[3975:90b] CoreData: sql: pragma freelist_count
2014-09-29 11:41:00.988 CoreDataTest[3975:90b] CoreData: annotation: sql execution
time: 0.0006s
2014-09-29 11:41:01.542 CoreDataTest[3975:90b] CoreData: sql: SELECT 0, t0.Z_PK,
t0.Z_OPT, t0.ZEMAIL, t0.ZNAME, t0.ZPHONE FROM ZSTUDENT t0
```