For this framework I have finish the storyboard skeleton（I have create some class for the views, but not all the views），three data model and a simple login (primary login). The login isn't finished, just use one textfiled(the upper one) for test.

You can type leon on the user name textfield. And click login. Then you will go to the primaryusermenu. Then click event, you will go to the event interface. There will be two event on it, just for test. Then you can tap add for add new event. Or tap each line to edit that event. **Swipe on a line to delete that event.**

I didn't set any layout for the storyboard. So what you see when you in storyboard is different from running time. You need to complete the autoLayout of your part.

Make sure that you adhere to the MVC(see lecture1). Which means all the manipulation of data is done by using data model functions, if you don't find the function you want in data model, then add some you self. But make sure the function you add is generic, not just for some particular use.

Modify you own branch of github. Never Never change the master branch without contacting other guys.

If you have any question when coding, please wechat me(Leon).

CLASS DESCREPTION

1. MNuser: this is the data model of the user. We distinguish primary user and assistant user by the property userType. And for this version we just assume that a primary user is only paired with one assistant user.
   So for each user. No matter it is a primary or assistant, we use his ID to fetch events from database, meanwhile, we use his paired user ID to fetch other events from database, these two part of events is the whole events witch associated with that user.
   When a event is edited, it's createdate is still the original createDate.

2. MNEvent: this is the data model of the user. the event has it creatorID and createDate, using this two properties we can find a particular event from the database.

3. MNEventlist: basically this is a encapsulation of NSMutableArray. This array stores all the event. I provide two initwith function to you(one for the future events another one for the past events). And in the header file, you can see some function declaration, this is the public functions you can use.

All the public functions you can use in declared in header file, just check them.

HOW TO READ AND WRITE CODE USING THIS FRAMWORK

1. the codes with the comment `//demo` is some demo for the using of some important method. Delete them when you code. All the important ios method is included in the demo code, And you can imitate them to do the detail job.
What you need to do is indicated by the comment within that code section.
For example:

```
- (IBAction)logInButton:(id)sender
{
    //demo
    //get the username and pin from database, if match performsegue,
or give user some alert.
    if([self.userNameTextField.text isEqualToString:@"hello"])
        [self performSegueWithIdentifier:@"logIn" sender:self];
    else
    {
        UIAlertView *alert = [[UIAlertView alloc]
                              initWithTitle:@"Error"
                              message:@"Your user ID or password is
invalid"
                              delegate:nil
                              cancelButtonTitle:@"OK"
                              otherButtonTitles:nil];
        [alert show];

    }
    //demo
}
```

in this function. I just compare the string of UserNameTextField with "hello"(just for test), if they equals, perform the segue, otherwise

just generate an alert view to tell the user. What you need to do is to make a query to the database and check whether this is a valid user and password, if it is then you call performSeguewithIdentifier to segue to the mainMenu, otherwise, give some alert to the user.

2. comment //??? Is the code you need to fulfill.

3. to make the textfield disappear after user type return. You need to add an uitextfileddelegate protocol in interface like this:

```
@interface PrimaryLoginViewController () <UITextFieldDelegate>
```

then add a method in the implementation like this:

```
//this method is to hide the textfield when the user type return
-(BOOL) textFieldShouldReturn:(UITextField *)textField
{
    [textField resignFirstResponder];
    return YES;
}
```

then add this code in viewDidLoad() like this:

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    self.userNameTextField.delegate = self.
}
```

remember that you need to set the self to be all the textfiled's delegate.

4. If you are confused which kind of segue you should use. Just use the modal segue. And remember to embed a navigatorconroller for each viewcontroller you segue to(if you use modal segue). And you can you unwind`````` function to exit a view(by control drag a button to the exit in the bottom bar) when you use modal segue. This hasn't been mentioned in the lectures, but you can find them in the framework, and hopefully you can figure out how the unwind``` functions working.

common errors

1. sometime the xcode editor can display the code, it's a bug, just reopen xcode. Or restart your computer. Or edit the code in other editor.
2. when you get error using tablecell, make sure you give the prototype cell the identifier.
3. When some function of a object doesn't perform as you want, check whether you

have allocate for it before using it;

4. Xcode 4.63 will sometimes shutdown in my computer. And maybe if it happened in your computer to, there may appear some code like this in the original code

```
=======
=======
>>>>>>> 9cf97bc9734ae8e3b289f7289c859c6cc202b363
=======
>>>>>>> 9cf97bc9734ae8e3b289f7289c859c6cc202b363
=======
>>>>>>> 9cf97bc9734ae8e3b289f7289c859c6cc202b363
just delete them.
```