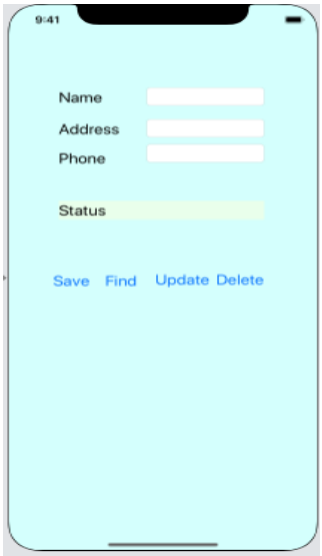
In-class Activity on FMDB app

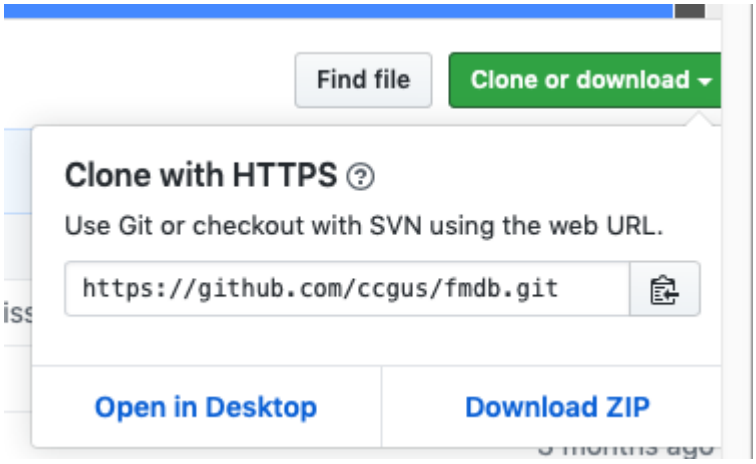
1. Create a Single View App with XCode and name it as **FMDBapp**. Add four labels, three textfields and four buttons (Save, Find, Update, Delete) to storyboard with a layout like this below:



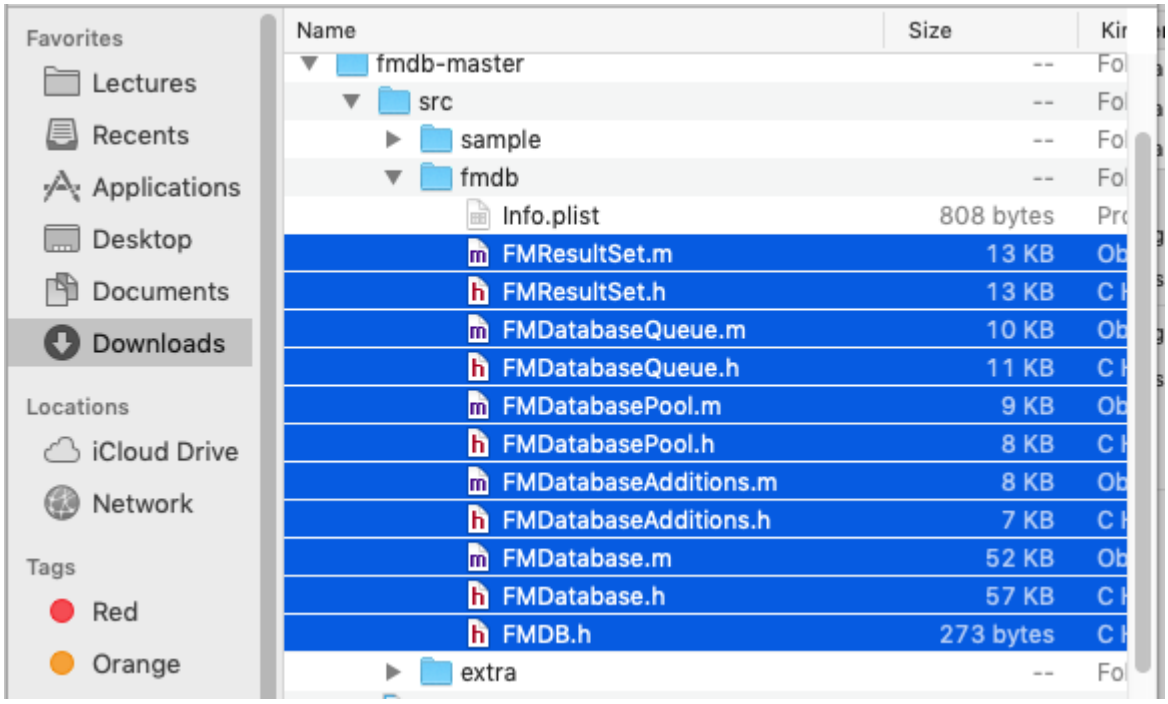
2. Connect the three textfields, the four buttons and the Status label to ViewController.swift file as new referencing outlets and IBActions. The file should look like this below:

```
import UIKit

class ViewController: UIViewController {
    @IBOutlet weak var name: UITextField!
    @IBOutlet weak var address: UITextField!
    @IBOutlet weak var phone: UITextField!
    @IBOutlet weak var status: UILabel!

    @IBAction func saveRecord(_ sender: Any) {
    }
    @IBAction func findRecord(_ sender: Any) {
    }
    @IBAction func updateRecord(_ sender: Any) {
    }
    @IBAction func deleteRecord(_ sender: Any) {
    }


    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
    }
}
```

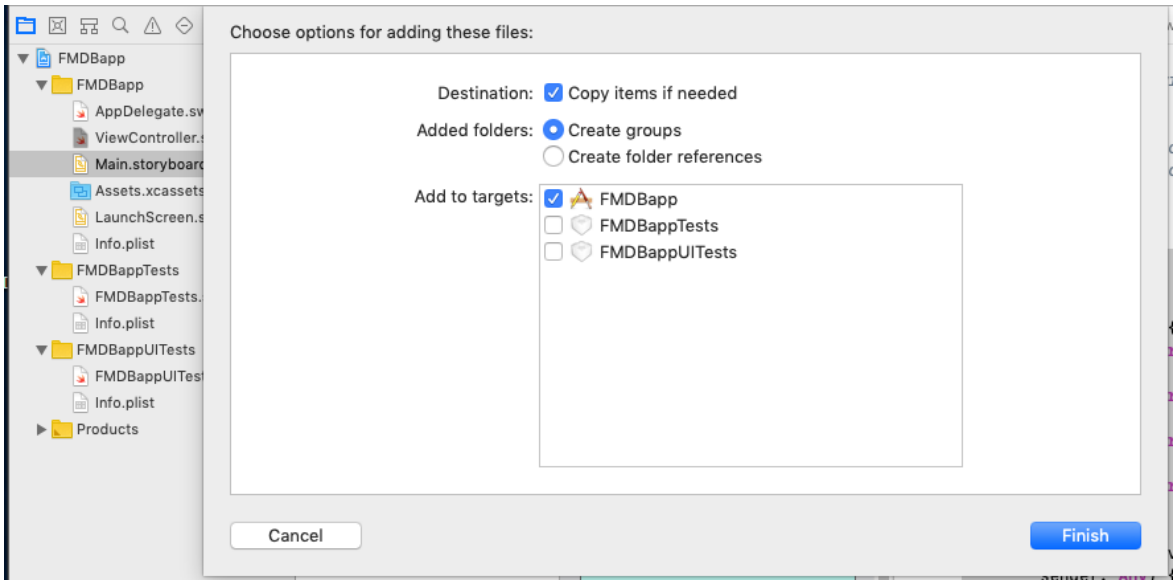
3. Next download FMDB wrapper program (written in Objective-C) from GitHub link below: https://github.com/ccgus/fmdb.git



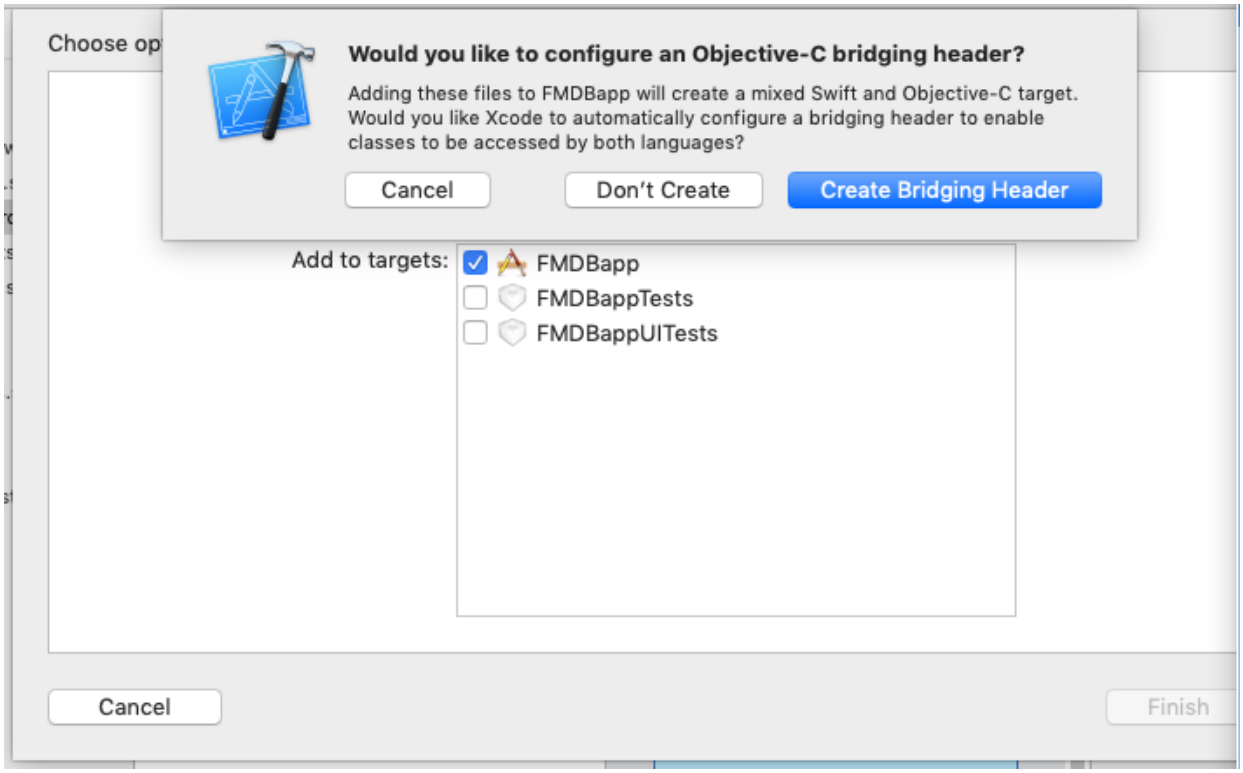
4. Unzip it, and then select and drag the **eleven** selected files into your **FMDBapp** project folder.



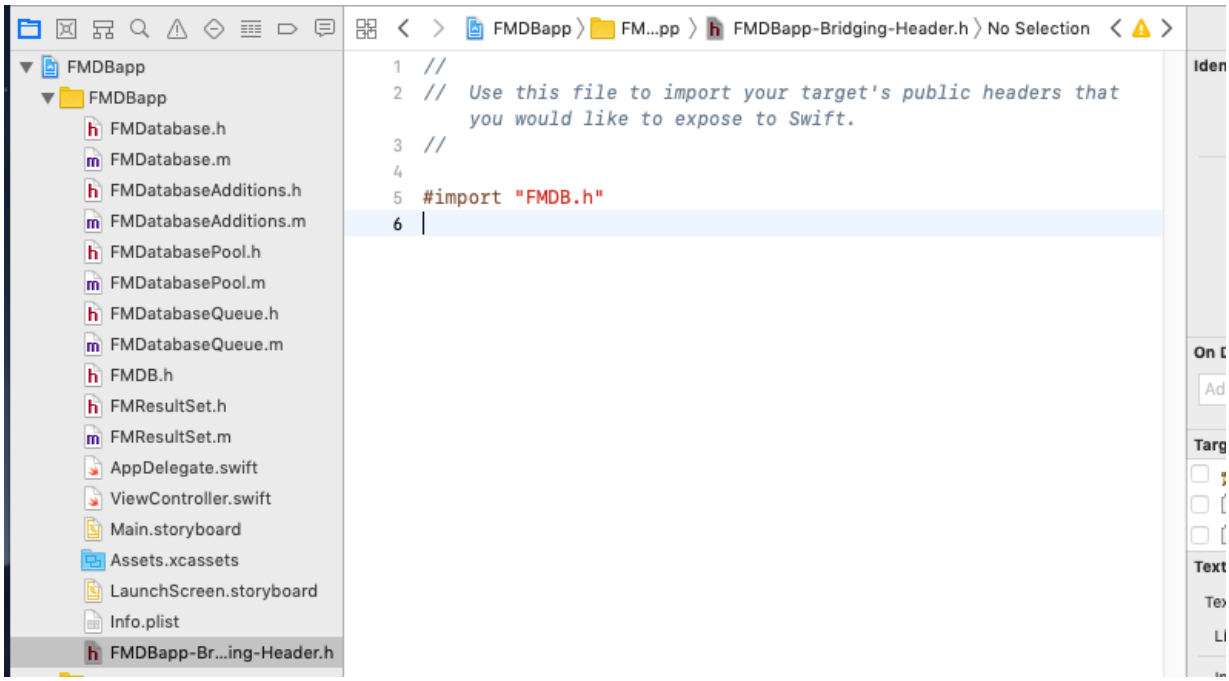
A popup window will appear like this:



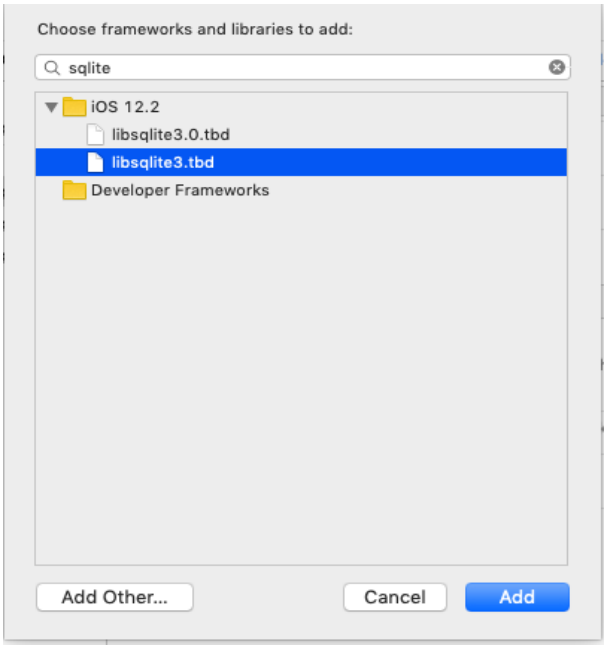
Check “Copy items if needed” check box → Finish → Create Bridging Header



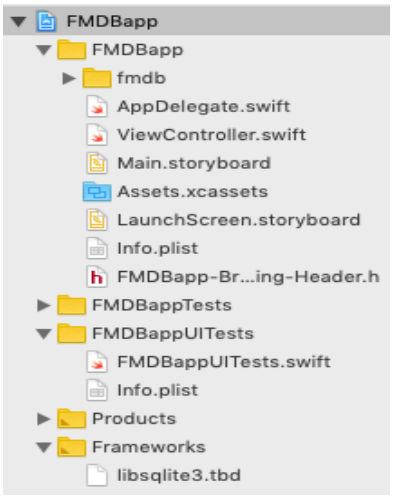
Next add an import statement like this into the header file (FMDBapp-Bridging-Header.h):
#import "FMDB.h"



5. Add SQLite library file to the project:
   Click on the project in Project Navigator → Build Phases → Expand “Link Binary With Libraries” → Click on the “+” → search sqlite → select **libsqlite3.tbd** → Add



Now the Frameworks folder should contains **libsqlite3.tdb** file:



6. Let's initialize a database file named "contacts.db" . If it already exits in the iOS device, the app will create an instance of it; otherwise, our app will create a empty database file to work with it. The app will open the database and make the table "CONTACTS" available. If the table doesn't exit, it will create the empty table to work with. The app will do all this in the ViewDidLoad () method in the ViewController class file with the following code:

```
var databasePath = String()

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.

        let filemgr = FileManager.default
        let dirPaths = filemgr.urls(for: .documentDirectory,  in: .userDomainMask)

        databasePath = dirPaths[0].appendingPathComponent("contacts.db").path

        if !filemgr.fileExists(atPath: databasePath as String) {

            let contactDB = FMDatabase(path: databasePath as String)

            if (contactDB.open()) {
                let sql_stmt = "CREATE TABLE IF NOT EXISTS CONTACTS (ID INTEGER
PRIMARY KEY AUTOINCREMENT, NAME TEXT, ADDRESS TEXT, PHONE TEXT)"
                if !(contactDB.executeStatements(sql_stmt)) {
                    print("Error: \(contactDB.lastErrorMessage())")
                }
                contactDB.close()
            } else {
                print("Error: \(contactDB.lastErrorMessage())")
            }
        }

    }
```

7. Complete the IBAction saveRecord() method with the following code:

```
        let contactDB = FMDatabase(path: databasePath as String)
        if (contactDB.open()){

            let insertSQL = "INSERT INTO CONTACTS (name, address, phone) VALUES
('\(name.text!)', '\(address.text!)', '\(phone.text!)')"

            let result = contactDB.executeUpdate(insertSQL, withArgumentsIn: [])

            if !result {
                status.text = "Failed to add contact"
                print("Error: \(contactDB.lastErrorMessage())")
            } else {
                status.text = "Contact Added"
                name.text = ""
                address.text = ""
                phone.text = ""
            }
        } else {
            print("Error: \(contactDB.lastErrorMessage())")
        }
```

8. Complete the IBAction findRecord() method with the following code:

```
        let contactDB = FMDatabase(path: databasePath as String)
        if (contactDB.open()) {
            let querySQL = "SELECT address, phone FROM CONTACTS WHERE name =
'\(name.text!)'"

            let results:FMResultSet? = contactDB.executeQuery(querySQL,  withArgumentsIn: [])

            if results?.next() == true {
                address.text = results?.string(forColumn: "address")
                phone.text = results?.string(forColumn: "phone")
                status.text = "Record Found"
            } else {
                status.text = "Record not found"
                address.text = ""
                phone.text = ""
            }
            contactDB.close()
        } else {
            print("Error: \(contactDB.lastErrorMessage())")
        }
```

9. Complete the IBAction updateRecord() method with the following code:

```
        let contactDB = FMDatabase(path: databasePath as String)
        if (contactDB.open()) {
            let updateSQL = "UPDATE CONTACTS set address = '\(address.text!)', phone =
'\(phone.text!)' WHERE name = '\(name.text!)'"

            let result = contactDB.executeUpdate(updateSQL, withArgumentsIn: [])

            if !result {
                status.text = "Failed to update contact"
                print("Error: \(contactDB.lastErrorMessage())")
            } else {
                status.text = "Contact Updated"
                name.text = ""
                address.text = ""
                phone.text = ""
            }
            contactDB.close()
        } else {
            print("Error: \(contactDB.lastErrorMessage())")
        }
```

10. Complete the IBAction deleteRecord() method with the following code:

```
        let contactDB = FMDatabase(path: databasePath as String)
        if (contactDB.open()) {
            let deleteSQL = "DELETE FROM CONTACTS WHERE name = '\(name.text!)'"

            let result = contactDB.executeUpdate(deleteSQL, withArgumentsIn: [])
            if !result {
                status.text = "Failed to delete contact"
                print("Error: \(contactDB.lastErrorMessage())")
            } else {
                status.text = "Contact deleted"
                name.text = ""
                address.text = ""
                phone.text = ""
            }
            contactDB.close()
        } else {
            print("Error: \(contactDB.lastErrorMessage())")
        }
```

11. Run the app and try out the save, find, update and delete operations. Take a few screen shots and submit them.