

Developer Manual

Toward TCC Version 1.1

**TEXAS CHRISTIAN UNIVERSITY
COMPUTER SCIENCE DEPARTMENT**

May 7, 2013

Authored by: Matthew Bauer, Reid Mulkey, Jose Segura

Developer Manual

Version 1.1

Revision Sign-off

By signing below, the team member certifies that he has read the entire document and has, to the best of his knowledge, found the information contained herein to be accurate and relevant within the document.

Name	Signature	Date
Matthew Bauer		
Reid Mulkey		
Jose Segura		

Developer Manual

Version 1.1

Revision History

The following is a history of revisions made to this document.

Document Version	Date Submitted	Changes
Developer Manual V1.0	04/27/2013	Initial Version, Skeleton Outline
Developer Manual V1.1	05/07/2013	Updates to text for Web and iOS Development

Developer Manual

Version 1.1

Table of Contents

REVISION SIGN-OFF	2
REVISION HISTORY	3
TABLE OF CONTENTS	4
1 INTRODUCTION	6
1.1 PURPOSE OF DOCUMENT	6
1.2 OVERVIEW	6
2 WEB APPLICATION	7
2.1 DEVELOPMENT ENVIRONMENT	8
2.2 TOWARD TCC WEB	10
2.2.1 MENU BAR	11
2.2.2 HOME SCREEN	12
2.2.3 FOOTER SECTION	13
2.3 CAREER PAGE	13
2.4 CHECKLIST	15
2.4.1 CHECKLIST GRADES	17
2.5 VIDEO PAGE	18
3 iOS APPLICATION	19
3.1 DEVELOPMENT ENVIRONMENT	19
3.2 CLASS STRUCTURE	20
3.2.1 CLASSES IN iOS	20
3.2.2 FILE STRUCTURE	20
3.2.3 IMPORTANT CLASSES	21
3.3 CHECKLIST	23
3.3.1 MODIFY CHECKLIST TEXT	23
3.3.2 ADD CHECKLIST TASKS	24
3.3.3 MODIFY CHECKLIST DATES	29
3.4 WEB PAGES	30
3.4.1 MODIFY EXISTING LINKS	30
3.4.2 ADD ADDITIONAL LINKS	31
4 ANDROID APPLICATION	34
4.1 DEVELOPMENT ENVIRONMENT	34
4.2 CLASS STRUCTURE	34
4.2.1 NAMING SYSTEM	34
4.2.2 IMPORTANT CLASSES	35
4.2.3 OBJECTS	37

Developer Manual

Version 1.1

4.3 CHECKLIST.....	38
4.3.1 MODIFY CHECKLIST TEXT	38
4.3.2 ADD CHECKLIST TASKS.....	38
4.3.3 MODIFY CHECKLIST DATES	39
4.4 WEB PAGES	40
4.4.1 MODIFY EXISTING LINKS	40
4.4.2 ADD ADDITIONAL LINKS.....	40
5 GLOSSARY OF TERMS	41

Developer Manual

Version 1.1

1 Introduction

1.1 Purpose of Document

The purpose of this document is to assist developers in modifying and deploying the Toward TCC application. This assistance is provided in the form of textual descriptions as well as screenshots of the application.

1.2 Overview

Section 2 assists the developer with the Toward TCC web application.

Section 3 assists the developer with the Toward TCC iOS application.

Section 4 assists the developer with the Toward TCC Android application.

Developer Manual

Version 1.1

2 Web Application

The website is composed of different components, including one folder of CSS files, one folder of images, and one folder of JS files, or JavaScript files. In Fig 2.1, you can see the breakdown of all the files that are available.

CSS files include the formatting of all the components for the website. If anything needs to be changed with colors or different fonts, you have to change them in the CSS folder. All files need to be updated to include the change.

The images folder (img) has all the images linked to the website. You can edit the contents of this folder, but please be cautious if you remove a file, it will no longer display on the website. Renaming an image can also cause issues with the layout, please use caution when changing image sizes and image names.

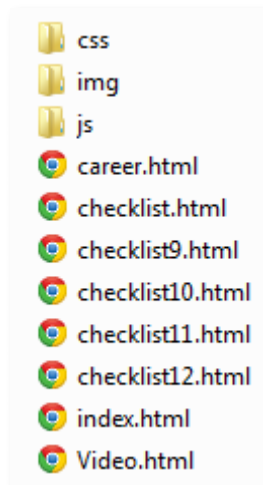


Fig 2.1

Developer Manual

Version 1.1

2.1 Development Environment

The web application was written using the Sublime Text 2 editor. Although it is not necessary for you to use this type of editor any editor that you feel comfortable with will suffice, eg., Notepad, Notepad++, WordPad.

If you want to use the Sublime Text editor, you can obtain it from <http://www.sublimetext.com/> and clicking on the download button in the menu bar. This has been demonstrated in Fig 2.1

After you have selected to download this program, you can go ahead and select the environment you are working in. For example, if you are using a Windows 64 bit, select that version in the list provided. This has been demonstrated in Fig 2.2 below. After the installation, open the sublime text program. Once the program has opened, Go to file > open folder > and open the folder that has the index.html file for this web application.

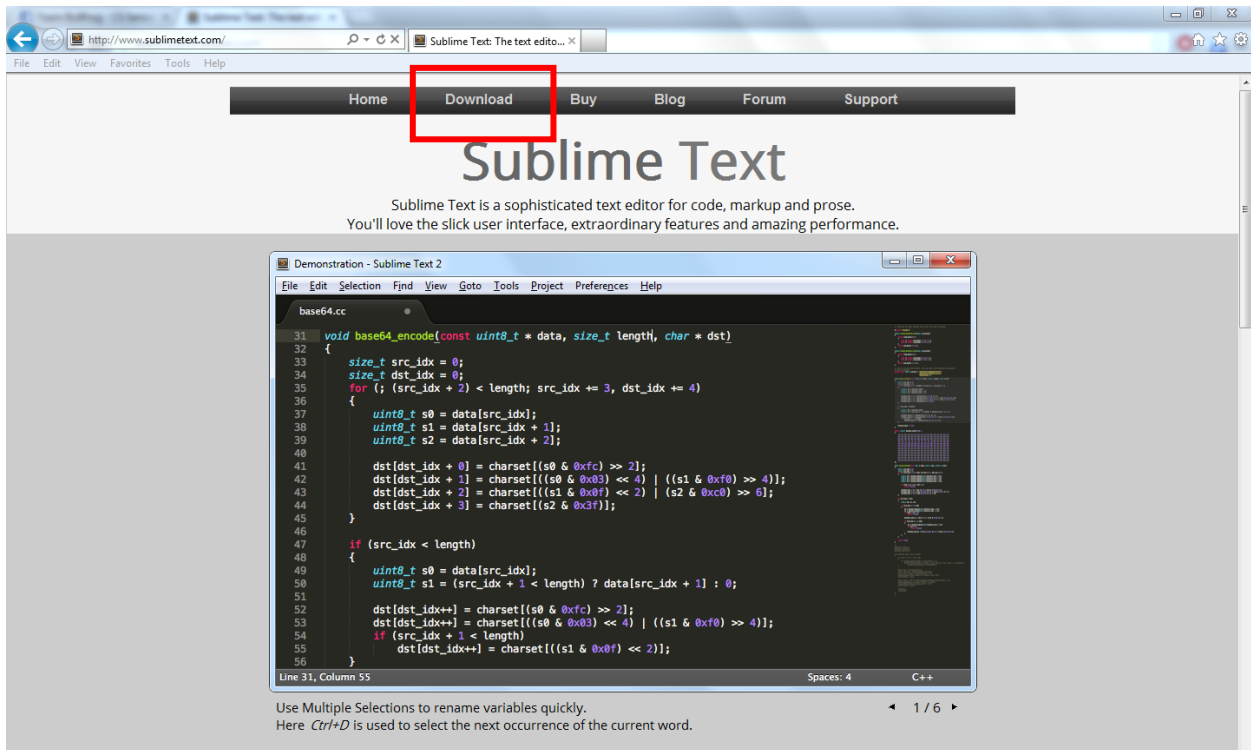


Fig 2.1

Developer Manual

Version 1.1

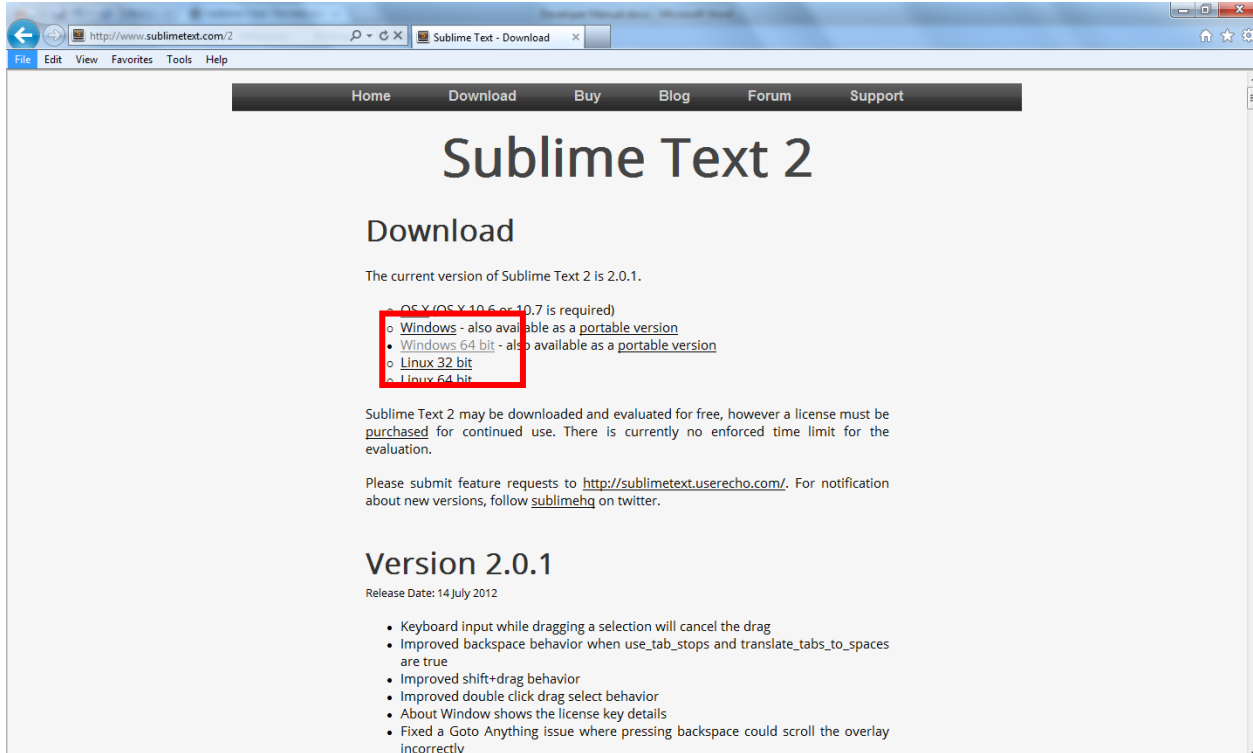


Fig 2.2

Developer Manual

Version 1.1

2.2.1 Menu Bar

The menu bar in Fig 2.5 is shown as seen from a browser window. The html code for the menu bar is shown in Fig 2.6. You can edit the content of the menu bar from this section in the index.html file. Each html file has its own menu bar. If a change is made on one html file, it should be updated on each one.



Fig 2.5

```
<!-- Menu Bar -->
<div class="navbar navbar-inverse navbar-fixed-top">
  <div class="navbar-inner">
    <div class="container-fluid">
      <a class="btn btn-navbar" data-toggle="collapse" data-target=".nav-collapse">
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </a>
      <!-- <a class="brand" href="index.html">Toward TCC</a> -->
      <!-- <a class="brand"></a> -->
      <div class="nav-collapse collapse">
        <ul class="nav">
          <li class="active"><a href="index.html">Home</a></li>
          <li><a href="http://www.tccd.edu/Admissions/Apply.html">Apply Now</a></li>
          <li><a href="career.html">Career</a></li>
          <li><a href="checklist.html">Checklist</a></li>
          <li><a href="Video.html">Videos</a></li>
        </ul>
      </div><!--/.nav-collapse -->
    </div>
  </div>
</div>
```

Fig 2.6

Developer Manual

Version 1.1

2.2.2 Home Screen

The body content can be modified in the body section in the index.html. You can modify content by removing and adding content in this section. Fig 2.7 shows you what the body section would look like in a browser window, and Fig 2.8 illustrates what you would see in an html editor.

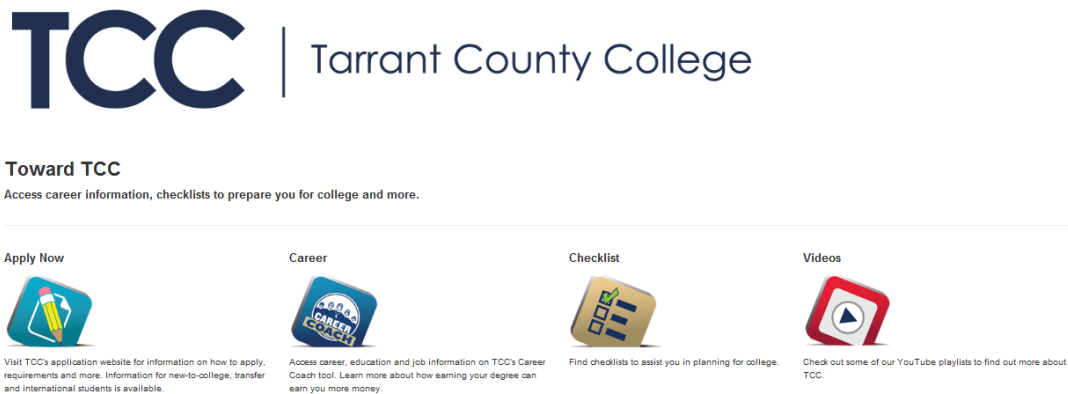


Fig 2.7

```
<div class="container-fluid">
  <div class="span1"></div>
  <div class="span 12">
    <!-- <div class="hero-unit"> -->
    <div>
      
      </br>
      </br>
      </br>
      </br>
      <h2>Toward TCC</h2>
      <h4>Access career information, checklists to prepare you for college and more.</h4>
    </div>
  </div>
</div>

<!-- 4 sections broken down -->

<div class="row-fluid">
  <div class="span3">
    <h4>Apply Now</h4>
    <p><a href="http://www.tccd.edu/Admissions/Apply.html"></a></p>
    <p>Visit TCC's application website for information on how to apply, requirements and more. Information for new-to-college, transfer and international students is available.</p>
    <!-- <p><a class="btn btn-warning" href="http://www.tccd.edu/Admissions/Apply.html">View details &raquo;</a></p> -->
  </div>
  <div class="span3">
    <h4>Career</h4>
    <p><a href="career.html"></a></p>
    <p>Access career, education and job information on TCC's Career Coach tool. Learn more about how earning your degree can earn you more money.</p>
    <!-- <p><a class="btn btn-inverse" href="career.html">View details &raquo;</a></p> -->
  </div>
  <div class="span3">
    <h4>Checklist</h4>
    <p><a href="checklist.html"></a></p>
    <p>Find checklists to assist you in planning for college.</p>
    <!-- <p><a class="btn btn-primary" href="checklist.html">View details &raquo;</a></p> -->
  </div>
  <div class="span3">
    <h4>Videos</h4>
    <p><a href="video.html"></a></p>
    <p>Check out some of our YouTube playlists to find out more about TCC.</p>
    <!-- <p><a class="btn btn-danger" href="video.html">View details &raquo;</a></p> -->
  </div>
</div>
</div>
```

Fig 2.8

Developer Manual

Version 1.1

2.2.3 Footer Section

In the footer section, you will have the copyright as well as the images for both the play stores. These images can be modified or they can be used as is. In Fig 2.9 you can see what the browser would look like when viewing the footer section. Fig 2.10 shows you what the editor section looks like when viewing from an editor. Each website has its own footer section that can be easily modified at the bottom of the html file.

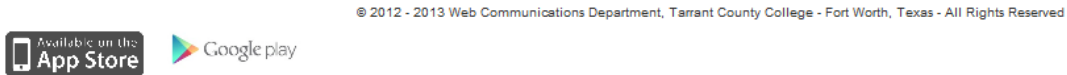


Fig 2.9

```
<!-- Footer section -->
<div id="footer">
  <div class="container">
    <footer>
      <p>© 2012 - 2013 Web Communications Department, Tarrant County College - Fort Worth, Texas - All Rights Reserved</p>
    </footer>
  </div>
</div>
</div>
<!-- App store icons -->
<div class="span12">
  <div class="span2">
    
  </div>
  <div class="span2">
    
  </div>
</div>
</div>
```

Fig 2.10

2.3 Career Page

The career page can be accessed through the career.html file and can easily be modified. The main body hosts two different sections in the career page that lists the two different websites that it accesses. The first is the career coach tool that is provided by TCC. This can be seen in Fig 2.13. You can edit the link as well as the amount of spacing in this snippet of code.

The second section is the GenTX.org section. This is a tool used to provide information about professional wages with education. The GenTX.org link can also be seen in Fig 2.13.

Developer Manual

Version 1.1

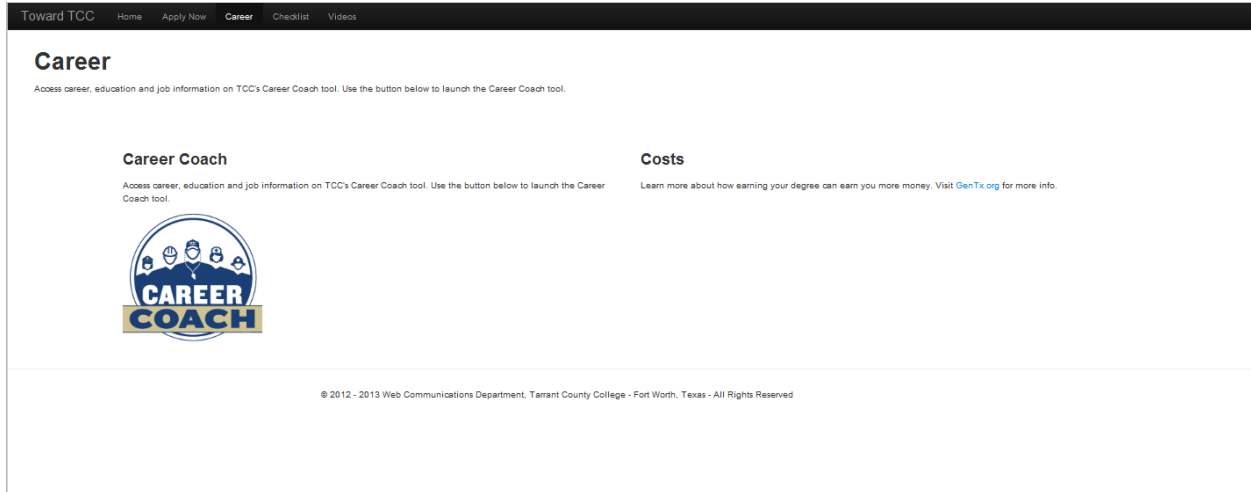


Fig 2.11

```
<body>
<div class="navbar navbar-inverse navbar-fixed-top">
  <div class="navbar-inner">
    <div class="container-fluid">
      <a class="btn btn-navbar" data-toggle="collapse" data-target=".nav-collapse">
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </a>
      <a class="brand" href="index.html">Toward TCC</a>
      <div class="nav-collapse collapse">
        <ul class="nav">
          <li><a href="index.html">Home</a></li>
          <li><a href="http://www.tccd.edu/Admissions/Apply.html">Apply Now</a></li>
          <li class="active"><a href="career.html">Career</a></li>
          <li><a href="checklist.html">Checklist</a></li>
          <li><a href="Video.html">Videos</a></li>
        </ul>
      </div><!--/.nav-collapse -->
    </div>
  </div>
</div>
<div class="container-fluid">
  <div class="span12">
    <h1>Career</h1>
    <p>Access career, education and job information on TCC's Career Coach tool. Use the button below to launch the Career Coach tool.</p>
    <hr />
  </div>
  <div class="row-fluid">
    <div class="span1"></div>
    <div class="span5">
      <h3>Career Coach</h3>
      <p>Access career, education and job information on TCC's Career Coach tool. Use the button below to launch the Career Coach tool.</p>
      <div class="span5">
        <h3>Costs</h3>
        <p>Learn more about how earning your degree can earn you more money. Visit <a href="http://gentx.org/get-inspired/learn-more-earn-more/">GenTx.org</a> for more info.</p>
        <!--  -->
      </div>
    </div></span1-->
  </div>
</div>
```

Fig 2.12

Developer Manual

Version 1.1

```
<div class="row-fluid">
  <div class="span1"></div>
  <div class="span5">
    <h3>Career Coach</h3>
    <p>Access career, education and job information on TCC's Career Coach tool. Use the button below to launch the Career Coach tool.</p>
    <p><a href="http://tccd.emsicareercoach.com"></a></p>
  </div>
  <div class="span5">
    <h3>Costs</h3>
    <p>Learn more about how earning your degree can earn you more money. Visit <a href="http://gentx.org/get-inspired/learn-more-earn-more/">GenTx.org</a> for more info.</p>
    <!--  -->
  </div>
  <div class="span1"></div>
</div><!--/span-->
</div>
```

Fig 2.13

2.4 Checklist

The checklist home screen (Fig 2.14) can be accessed through the checklist.html file. The body of the file, as seen in Fig 2.15, can be easily modified to add multiple elements. Each button opens a particular grade level website. Checklist9.html is an example of what the naming convention for the grade level checklists is.

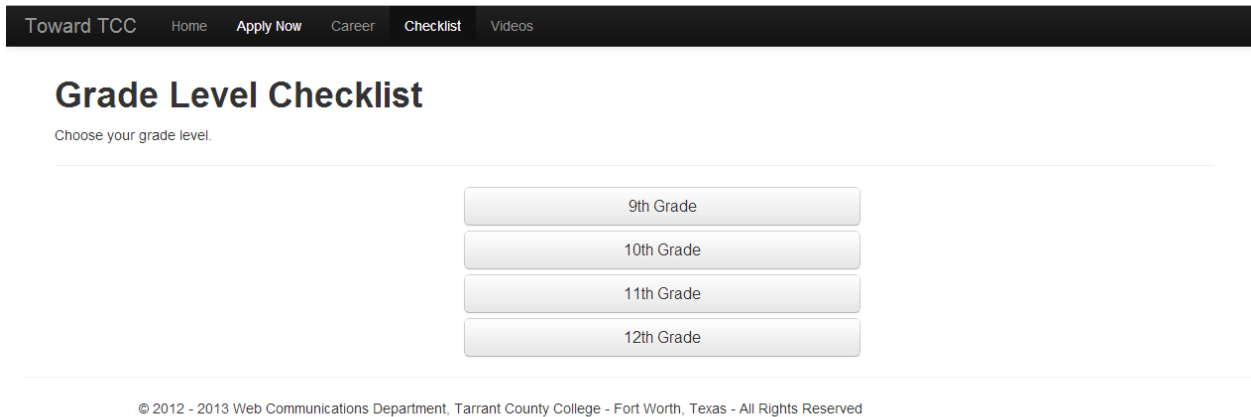


Fig 2.14

Developer Manual

Version 1.1

```
<div class="container-fluid">
  <div class="span12">
    <h1>Grade Level Checklist</h1>
    <p>Choose your grade level.</p>
    <hr />
  </div>
</div>

<div class="container-fluid">
  <div class="row-fluid">
    <div class="span12">
      <div class="span4"></div>
      <div class="span4 offset 4" style="max-width: 400px; margin: 0 auto 00px;">
        <a href="checklist9.html" class="btn btn-large btn-block">9th Grade</a>
        <a href="checklist10.html" class="btn btn-large btn-block">10th Grade</a>
        <a href="checklist11.html" class="btn btn-large btn-block">11th Grade</a>
        <a href="checklist12.html" class="btn btn-large btn-block">12th Grade</a>
      </div>
    </div>
  </div>
</div>
```

Fig 2.15

Developer Manual

Version 1.1

2.4.1 Checklist Grades

Fig 2.16 is an example of how a check list is viewed in a browser. This is the common format for all the 9th-12th grade checklists. The checklist9.html is seen in Fig 2.17. Each grade level has a similar format and is layered as an unordered html list. These can be easily modified to add additional information or to remove information in blocks. Unordered lists use CSS styling and are very compact, which will assist in screen resolution and screen size.

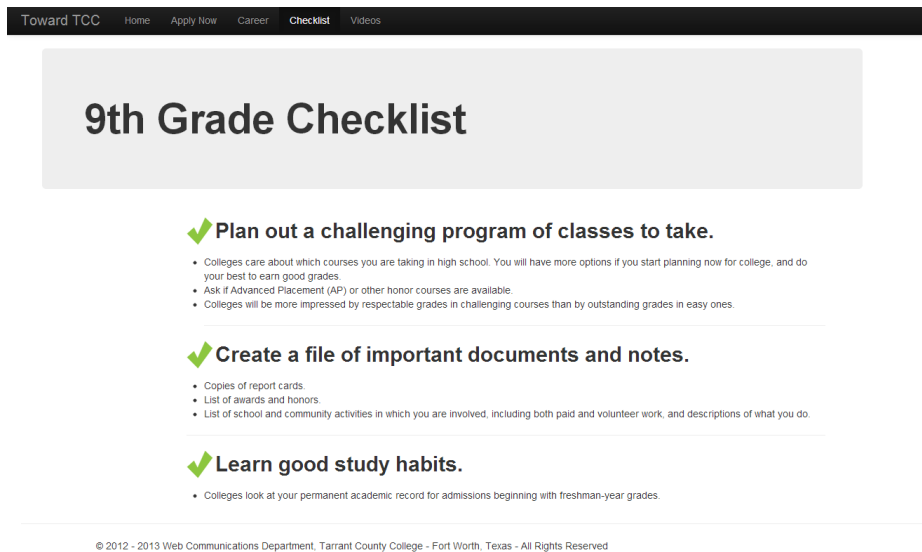


Fig 2.16

```
<!-- 9th grade checklist information -->
<div class="container-fluid">
  <div class="row-fluid">
    <div class="span12">
      <div class="span2"></div>
      <div class="span8 offset 2">
        <h2>Plan out a challenging program of classes to take.</h2>
        <ul>
          <li>Colleges care about which courses you are taking in high school. You will have more options if you start planning now for college, and do your best to earn good grades.</li>
          <li>Ask if Advanced Placement (AP) or other honor courses are available.</li>
          <li>Colleges will be more impressed by respectable grades in challenging courses than by outstanding grades in easy ones.</li>
        </ul>
        <hr />
        <h2>Create a file of important documents and notes.</h2>
        <ul>
          <li>Copies of report cards.</li>
          <li>List of awards and honors.</li>
          <li>List of school and community activities in which you are involved, including both paid and volunteer work, and descriptions of what you do.</li>
        </ul>
        <hr />
        <h2>Learn good study habits.</h2>
        <ul>
          <li>Colleges look at your permanent academic record for admissions beginning with freshman-year grades.</li>
        </ul>
      </div>
    </div>
  </div>
</div>
```

Fig 2.17

Developer Manual

Version 1.1

2.5 Video Page

The videos page can be accessed through the video.html file. Fig 2.18 shows what you would see through a web browser. Fig 2.19 shows you the body of the file. Each button is linked to a YouTube playlist, which can be easily managed through YouTube.

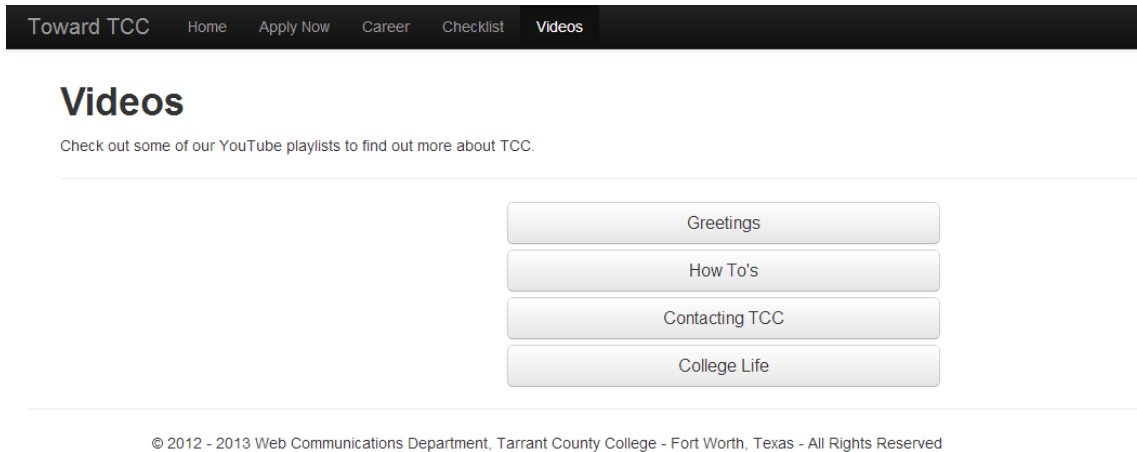


Fig 2.18

```
<!-- Video content body -->
<div class="container-fluid">
  <div class="span12">
    <h1>Videos</h1>
    <p>Check out some of our YouTube playlists to find out more about TCC.</p>
    <hr />
  </div>
</div>
<div class="container-fluid">
  <div class="row-fluid">
    <div class="span12">
      <div class="span4"></div>
      <div class="span4 offset 4" style="max-width: 400px; margin: 0 auto 00px;">
        <a href="http://www.youtube.com/playlist?list=PLMi1RxsDDWS_j5-c9NktWuLQ4YA7me1pa&feature=g-list" class="btn btn-large btn-block">Greetings</a>
        <a href="http://www.youtube.com/playlist?list=PLMi1RxsDDWS8-ET7sa_IQ9uALw8Jgn1Zj&feature=g-list" class="btn btn-large btn-block">How To's</a>
        <a href="http://www.youtube.com/playlist?list=PLMi1RxsDDWS-OguIzmggPC8buAdk7Asg&feature=g-list" class="btn btn-large btn-block">Contacting TCC</a>
        <a href="http://www.youtube.com/playlist?list=PLMi1RxsDDWS82Qq_s426Y4_MIUjkydWTb&feature=g-list" class="btn btn-large btn-block">College Life</a>
      </div>
    </div>
  </div>
</div>
```

Fig 2.19

Developer Manual

Version 1.1

3 iOS Application

3.1 Development Environment

The iOS application was written using Xcode 4.6.2 on an iMac using OS X 10.8.3. In order to install Xcode 4.6.2, visit <https://developer.apple.com/downloads/index.action#>. The download will require you to log in with an apple developer account. Search “Xcode” in the search box to the top left, and click on the dropdown menu for Xcode 4.6.2. Now click the link shown in Fig 3.1.

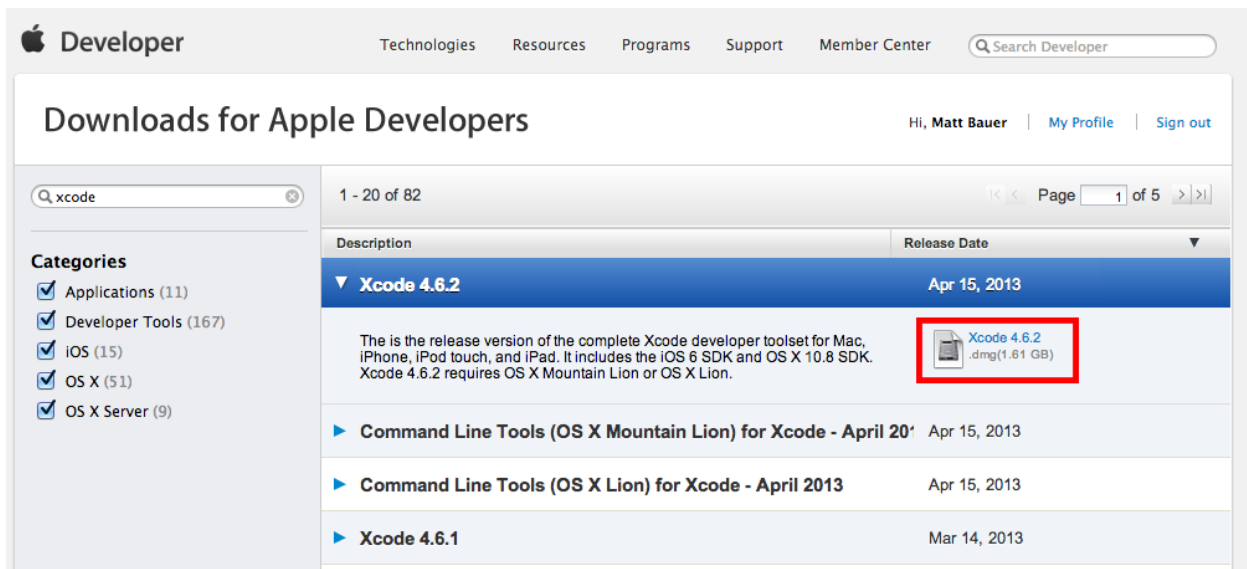


Fig 3.1

Once the download is complete, double click on the file. The files will be installed, and you will be taken to a screen to move Xcode into your Applications folder, shown in Fig 3.2. Drag Xcode icon and put it into your Applications folder.



Fig 3.2

Developer Manual

Version 1.1

After installation, open up Xcode and click on the “Open Other...” button on the bottom left of the screen. A file chooser will allow you to select a project to open. Select the “Toward TCC” project folder.

3.2 Class structure

3.2.1 Classes in iOS

All of the classes are associated with a certain screen in the application. There are also 2 classes for each screen: a header file and a main file. The header file contains the definitions for all the variables and methods the main file uses, and the main file contains all of the code.

3.2.2 File Structure

The variables that the file structure stores are shown and explained in Fig 3.3.

▼ Root	Dictionary	(4 items)
rememberedUsername	String	example_username
rememberMe	Boolean	YES
currentUser	Number	-1
▼ users	Array	(1 item)
▼ Item 0	Dictionary	(8 items)
▼ checklistOptions	Array	(4 items)
Item 0	Boolean	NO
Item 1	Boolean	NO
Item 2	Boolean	NO
Item 3	Boolean	NO
▼ checklistData	Array	(4 items)
▼ Item 0	Array	(3 items)
Item 0	Boolean	NO
Item 1	Boolean	NO
Item 2	Boolean	NO
▶ Item 1	Array	(5 items)
▶ Item 2	Array	(7 items)
▶ Item 3	Array	(5 items)
passwordHash	String	a password that has been hashed
securityQuestionAnswerHash	String	a security question that has been hashed
securityQuestion	String	users security question
username	String	example_username
gradeLevel	Number	-1
givenGradeReminder	Number	-1

Fig 3.3

rememberedUsername – Username the application is remembering.

rememberMe – Stores whether or not the application is remembering a username.

currentUser – The index into the users array of who the current user is. If it is -1, then no one is logged in.

users – The array that stores users information.

Developer Manual

Version 1.1

checklistOptions – Array storing the users reminder settings. Index 0 is for 1 week reminders, index 1 is for 2 week reminders, etc.

checklistData – Array storing the users checklist progress, with each year having an array of Boolean values representing if checkboxes have been filled in. Index 0 is 9th grade progress, index 1 is 10th grade progress, index 2 is 11th grade progress, and index 3 is 12th grade progress. Each year's array has as many Booleans as it has checkboxes.

passwordHash – This is a string that stores the result of hashing the user's password using the SHA1 hashing algorithm.

securityQuestionAnswerHash - This is a string that stores the result of hashing the user's security question answer using the SHA1 hashing algorithm.

securityQuestion – This is a string that stores the user's security question.

username – This is a string that stores the user's username.

gradeLevel – This number stores the user's grade level.

givenGradeReminder – This number stores the date the last grade reminder was given to the user, that reminds the user to change their grade when the new academic year comes along.

3.2.3 Important Classes

3.2.3.1 ViewController.m

This class is the class that controls the home screen. Listed here is a list of the main methods, and a summary of what they do. More in depth information can be found in the comments in the code.

-(void)viewWillAppear:(BOOL)animated – This method will check to make sure the file structure has been initialized on the device.

-(void)toast:(NSString*)toastString – This method will display a toast notification on the screen, with the toastString as the message.

-(IBAction)loginButtonPressed:(id)sender – This method is called when the login button is pressed. It will check credentials, change the visibility of account management objects, and give any reminders needed.

-(IBAction)logoutButtonPressed:(id)sender - This method is called when the log out button is pressed. It will reset the home screen objects so that a user is not logged in, it will log them out of the file structure, set the

Developer Manual

Version 1.1

remembered username if needed, and give the user a toast notification informing them they have been logged out.

-(void)changeOutletVisibility:(bool)visibility - This method will change the visibility of the account management objects. If visibility is YES, the logged out objects become visible, and the logged in objects become invisible. If visibility is NO, the logged in objects become visible, and the logged out objects become invisible.

-(void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender – This method is called when the screen is going to change. For the purposes of this application, this method only performs a task when transitioning to the WebPageViewController.m class/screen. This class will tell the next class what web page the user wants to visit, and then transition to that page.

3.2.3.2 WebPageViewController.m

This class will display all external web pages. Listed here is a list of the main methods, and a summary of what they do. More in depth information can be found in the comments in the code.

-(IBAction)backButtonPressed:(id)sender – This method is called when the back button is pressed. If there is a page to return to, the application will go back one page.

-(IBAction)forwardButtonPressed:(id)sender – This method is called when the forward button is pressed. If there is a page to move forward to, the application will go forward one page.

-(IBAction)refreshButtonPressed:(id)sender – This method is called when the refresh button is pressed. It will reload the page the browser is currently displaying.

-(void)setPage:(NSString*)jincPage – This method is called by other classes trying to view a website. In the -(void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender method of the preceding class, it will create a WebPageViewController.m object and set the page string to its desired website.

-(void)viewDidLoad – This method will set the browser's URL based off what page string was given to this class.

-(void)webView:(UIWebView *)webView didFailLoadWithError:(NSError *)error – This method is called when the web browser encounters an error. If the internet connection is lost, then the application will return to the previous screen and notify the user of the error.

Developer Manual

Version 1.1

3.3 Checklist

3.3.1 Modify Checklist text

In order to change text on the checklist screens, double click on one of the text views with text to be changed. For example, to demonstrate correcting a typo on the 9th grade checklist screen, item number #2, and hyphen number 3. First go to the “MainStoryboard_iPhone.storyboard” file, and navigate to the screen to be changed. This hyphenated text is not visible so move the scroll view content down. Click and drag on the scroll view (Fig 3.4) object and move it up (Fig 3.5). Then, expand it down so that the text is visible (Fig 3.6). Then, double click on the text view that needs to be changed (Fig 3.7), and change the text.

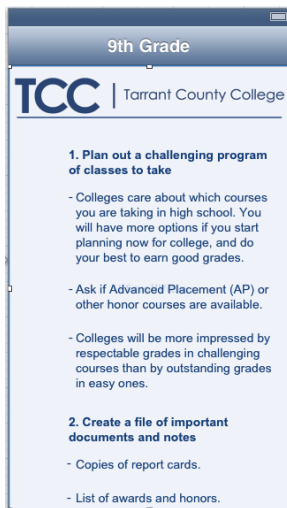


Fig 3.4

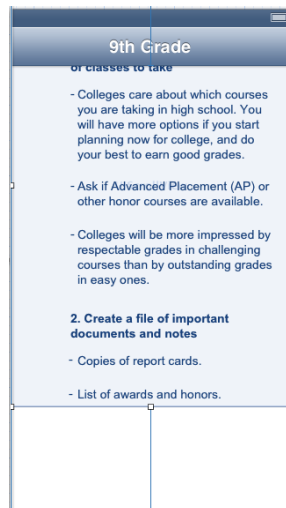


Fig 3.5

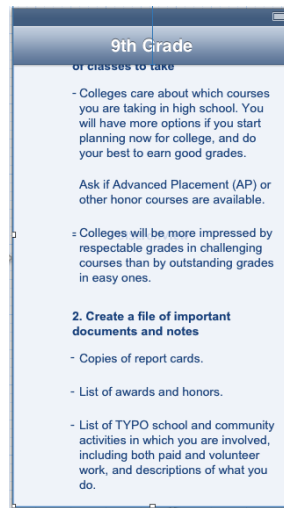


Fig 3.6

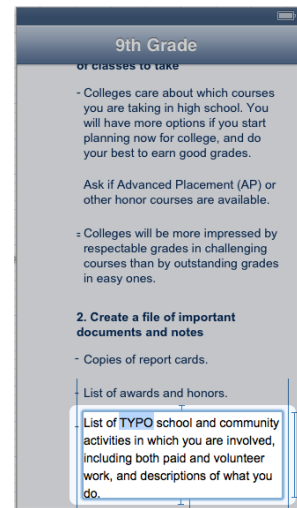


Fig 3.7

Make sure to change the scroll view back to how it was before implementing this change, or else it will not scroll properly. The default x and y positions should be 0. For small iPhones, the width should be 320 and the height should be 416. For the large iPhone view, the width should be 320 and the height should be 504. For iPads, the width should be 768 and the height should be 960. Make sure to modify both the iPhone and iPad storyboards. For an easy way to modify positioning, click on the 5th tab of the editor to the right, and modify the values under the view section, highlighted in Fig 3.8.

Developer Manual

Version 1.1

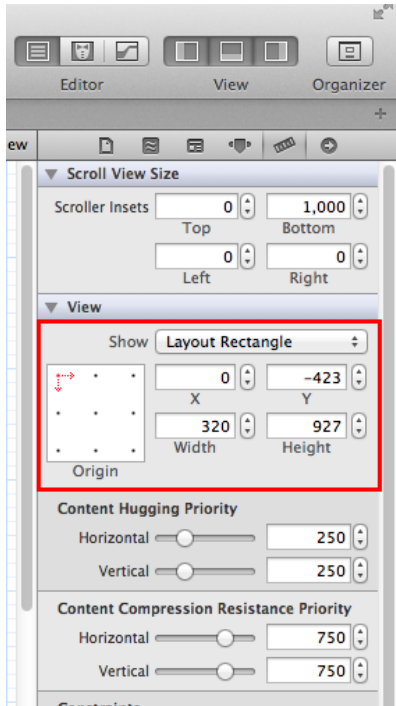


Fig 3.8

3.3.2 Add Checklist tasks

It is recommended not to add checklist tasks when this application is in production, for it will require users to reinstall in order to take advantage of the new checklist tasks. For this example, a task will be added to the iPhone 9th grade checklist screen, in between the 2nd and 3rd tasks. When new tasks are to be added to the project, make sure to do so on both iPhone and iPad.

3.3.2.1 Reposition the scroll view

In order to add a new task to a checklist, move the scroll view in such a way that the point to insert is visible. In order to move the scroll view, see Section 3.3.1 and Figs 3.4-3.6.

3.3.2.2 Duplicate a checklist task

In order to make sure the new checklist task to be inserted follows the same format established by the other tasks, left click out in empty space and drag a box around all of the components, and release the left click. Then, press command + c, then left click into empty space, and press command + v. This will generate a new task.

Developer Manual

Version 1.1

3.3.2.3 Add new task text

Change the heading and numbering of the new checklist task by double clicking on the heading label, and then edit the hyphenated text in a similar manner. If the new checklist task needs more hyphenated text, form a box around the hyphen and the text to the right of the hyphen, similar to 3.3.2.2, and press command + c and then command + v. Make sure to change the numbering after this task to reflect the changes. Now you should be at the point shown in Fig 3.9.

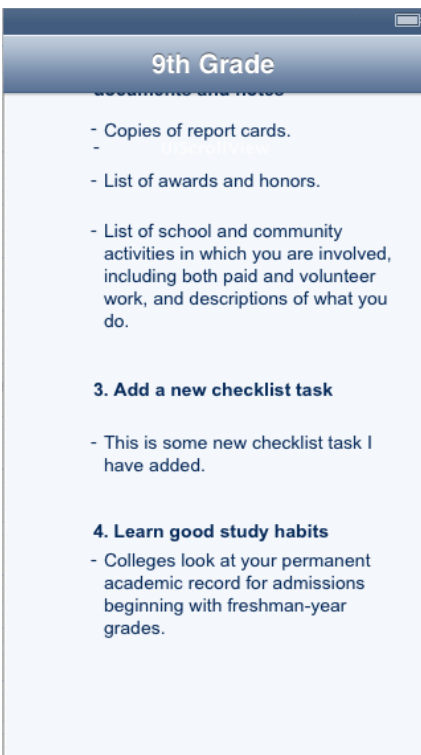


Fig 3.9

3.3.3.4 Ties to the code

The program will need to know what new task has been added, so it knows what text box is associated with the user's data. First, the definitions of the checkboxes must be changed in the "XGradeViewController.h" class. The checklist tasks defined there currently occur in numeric order. By adding a new number to the checklist, this will disrupt the file structure of people who were using the application before the task was added. For this example, a new checkbox definition will need to be added for the "NinthGradeViewController.h" (Fig 3.10). This new checkbox must be synthesized in "NinthGradeViewController.m" (Fig 3.11). Then a new "on click" method must be made for this checkbox (Fig 3.12) and it must be modified like all of the other checkboxes in "viewDidLoad" (Fig 3.13).

Developer Manual

Version 1.1

```
@interface NinthGradeViewController : UIViewController
{
    int currentUser;

    NSString* path;

    NSMutableDictionary* fileStruct;
    NSMutableDictionary* user;

    NSMutableArray* users;
    NSMutableArray* checklistData;
    NSMutableArray* gradeData;

    IBOutlet UIButton* checkbox1;
    IBOutlet UIButton* checkbox2;
    IBOutlet UIButton* checkbox3;
    IBOutlet UIButton* checkbox4;

    IBOutlet UIScrollView* scroller;
}
@property(nonatomic, strong) NSString* path;

@property(nonatomic, strong) NSMutableDictionary* fileStruct;
@property(nonatomic, strong) NSMutableDictionary* user;

@property(nonatomic, strong) NSMutableArray* users;
@property(nonatomic, strong) NSMutableArray* checklistData;
@property(nonatomic, strong) NSMutableArray* gradeData;

@property(nonatomic, strong) IBOutlet UIButton* checkbox1;
@property(nonatomic, strong) IBOutlet UIButton* checkbox2;
@property(nonatomic, strong) IBOutlet UIButton* checkbox3;
@property(nonatomic, strong) IBOutlet UIButton* checkbox4;

@property(nonatomic, strong) IBOutlet UIScrollView* scroller;

-(void)checkboxPressed:(NSNumber*)index isChecked:(bool)isChecked;
@end
```

Fig 3.10

```
@interface NinthGradeViewController ()
@end

@implementation NinthGradeViewController
@synthesize path;
@synthesize fileStruct, users, user, checklistData, gradeData;
@synthesize checkbox1, checkbox2, checkbox3, checkbox4;
@synthesize scroller;
```

Fig 3.11

```
-(IBAction)checkbox4Pressed:(id)sender
{
    if(currentUser < 0) return;
    [self checkBoxPressed:[NSNumber alloc]
        initWithInt:3] isChecked:![gradeData
        objectAtIndex:3] boolValue];
    [self setCheckBox:checkbox4 atIndex:[NSNumber
        alloc] initWithInt:3];
}
```

Fig 3.12

```
-(void)viewDidLoad
{
    //This block of code will gather relevant information from the file struct
    NSArray *documentPaths = NSSearchPathForDirectoriesInDomains(NSDocumentDir,
    path = [[documentPaths objectAtIndex:0] stringByAppendingFormat:@"%Users.p
    fileStruct = [[NSMutableDictionary alloc] initWithContentsOfFile:path];
    currentUser = [[fileStruct objectForKey:@"currentUser"] intValue];

    //if a user is logged on
    if(currentUser >= 0)
    {
        //then set the checkboxes according to their progress data
        users = [fileStruct objectForKey:@"users"];
        user = [users objectAtIndex:currentUser];
        gradeData = [[user objectForKey:@"checklistData"] objectAtIndex:0];

        [self setCheckBox:checkbox1 atIndex:[NSNumber alloc] initWithInt:0];
        [self setCheckBox:checkbox2 atIndex:[NSNumber alloc] initWithInt:1];
        [self setCheckBox:checkbox3 atIndex:[NSNumber alloc] initWithInt:2];
        [self setCheckBox:checkbox4 atIndex:[NSNumber alloc] initWithInt:3];
    }
    else //if a user is not logged on
    {
        //then hide all of the checkboxes
        [checkbox1 setHidden:YES];
        [checkbox2 setHidden:YES];
        [checkbox3 setHidden:YES];
        [checkbox4 setHidden:YES];
    }
}
```

Fig 3.13

If a change is being made in production, a try-catch statement must be added to avoid the program from crashing, shown in Fig 3.14. The try statement will detect if the application will crash when it accesses index 3 into the grade information. If it does, then there is an index out of bounds exception, so it will go to the catch statement, which will add an object to the spot the checklist task is being added, and move the ones after it forward by one. Then, the changes should be saved back to the file structure.

Developer Manual

Version 1.1

```
@try
{
    [self setCheckBox:checkbox1 atIndex:[NSNumber alloc] initWithInt:0]];
    [self setCheckBox:checkbox2 atIndex:[NSNumber alloc] initWithInt:1]];
    [self setCheckBox:checkbox3 atIndex:[NSNumber alloc] initWithInt:2]];
    [self setCheckBox:checkbox4 atIndex:[NSNumber alloc] initWithInt:3]];
}
@catch (NSEException *exception)
{
    [gradeData insertObject:[NSNumber alloc] initWithBool:NO] atIndex:2];

    [self setCheckBox:checkbox1 atIndex:[NSNumber alloc] initWithInt:0]];
    [self setCheckBox:checkbox2 atIndex:[NSNumber alloc] initWithInt:1]];
    [self setCheckBox:checkbox3 atIndex:[NSNumber alloc] initWithInt:2]];
    [self setCheckBox:checkbox4 atIndex:[NSNumber alloc] initWithInt:3]];

    [[fileStruct writeToFile:path atomically:YES];
}
}
```

Fig 3.14

Connections must be reset from the “MainStoryboard_iPhone.storyboard” and “MainStoryboard_iPad.storyboard” files, so the interface knows what checkboxes the code is now referring to. In order to do this, press the dark blue bar at the top of the application, and change the checkbox connections in the 6th tab of the editor to the right (Fig 3.15). You will select the checkbox attribute in the connections, and then remove their current “Touch Up Inside” and referencing outlets connections by clicking the “X” in the upper left corner. Now that they have been removed, they can reset by redrawing lines for both “Touch Up Inside” (Fig 3.16) and “checkboxX” (Fig 3.17) connections.

Developer Manual

Version 1.1

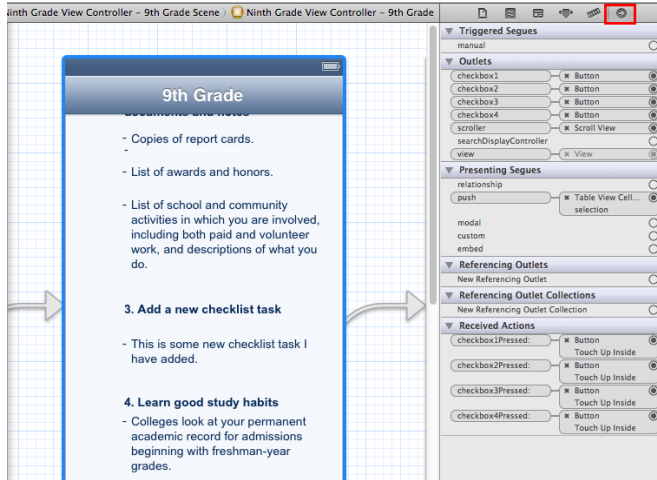


Fig 3.14

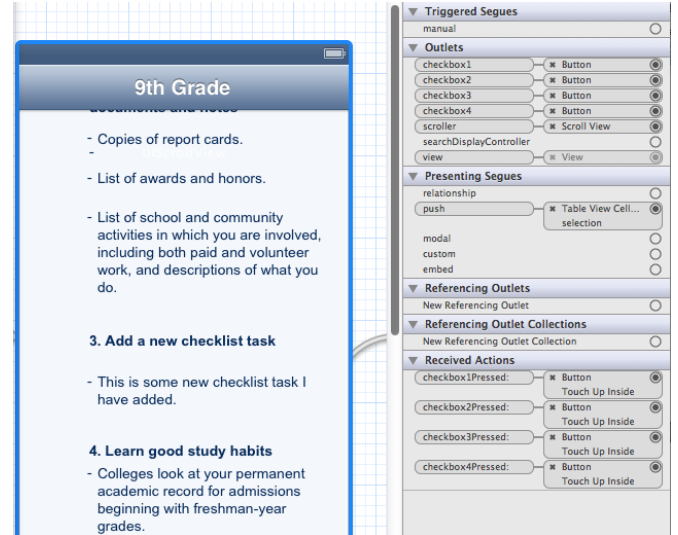


Fig 3.15

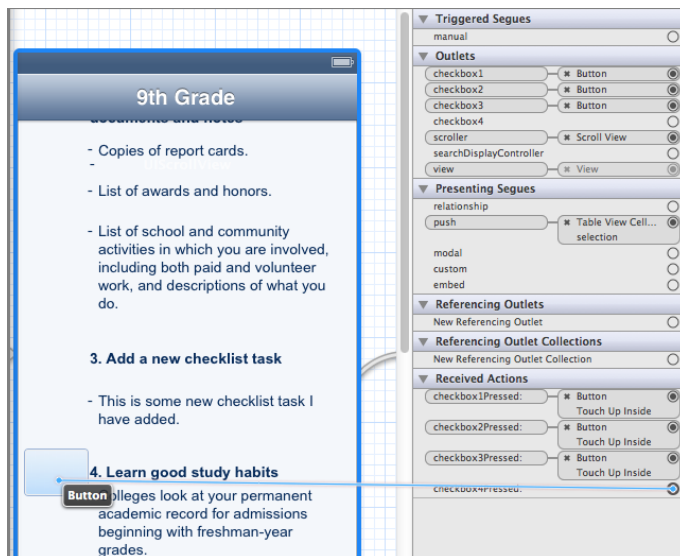


Fig 3.16

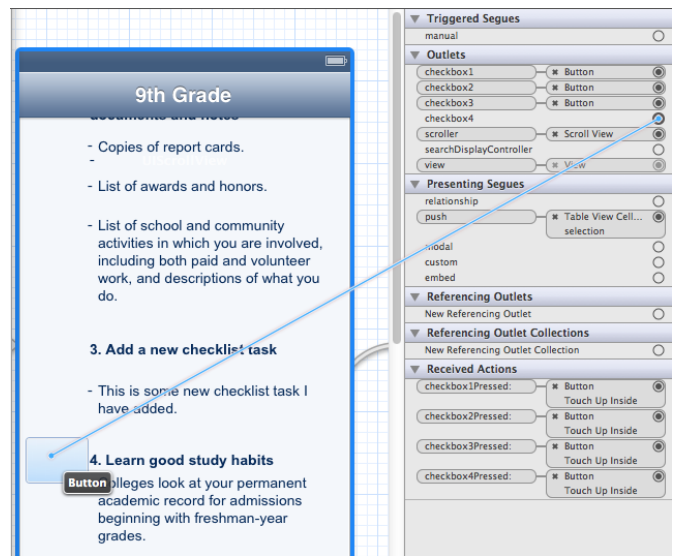


Fig 3.17

For the last step in adding a checklist task, go to the “submitButtonPressed” method within “CreateProfileViewController.m”. Here you will find 4 for loops (one for each year) that populate a new user’s checklist progress data (Fig 3.18). For this example, the number of checklist tasks for 9th grade is increased by 1, so the upper bound of the for loop must be increased by 1, shown in Fig 3.19.

Developer Manual

Version 1.1

```
for(int i = 0; i < 3; i++)
    [year1 addObject:[NSNumber alloc] initWithBool:NO]];
    unchecked.
for(int i = 0; i < 5; i++)
    [year2 addObject:[NSNumber alloc] initWithBool:NO]];
    unchecked.
for(int i = 0; i < 7; i++)
    [year3 addObject:[NSNumber alloc] initWithBool:NO]];
    unchecked.
for(int i = 0; i < 5; i++)
    [year4 addObject:[NSNumber alloc] initWithBool:NO]]; //s
    unchecked.
```

Fig 3.18

```
for(int i = 0; i < 4; i++)
    [year1 addObject:[NSNumber alloc] initWithBool:NO]];
    unchecked.
for(int i = 0; i < 5; i++)
    [year2 addObject:[NSNumber alloc] initWithBool:NO]];
    unchecked.
for(int i = 0; i < 7; i++)
    [year3 addObject:[NSNumber alloc] initWithBool:NO]];
    unchecked.
for(int i = 0; i < 5; i++)
    [year4 addObject:[NSNumber alloc] initWithBool:NO]]; //s
    unchecked.
```

Fig 3.19

3.3.3 Modify Checklist dates

In order to modify checklist dates, go to the “loginButtonPressed” method inside of “ViewController.m”. Here, several integer arrays are defined (Fig 3.20). Each grade level has 2 different arrays associated with it: XGradeMonths and XGradeDueDates. XGradeMonths is ordered from lowest to highest month for each task, and XGradeDueDates is the order the months of the tasks appears on the checklist page. In order to modify the January task of 12th grade to February, simply remove 1 from both arrays and put 2 in its place. In order to enable 9th or 10th grade reminders, remove the comments before their array definitions and surrounding their code. There are additional comments found in the code to aid in this process.

```
//int ninthGradeMonths[5] = {1,3,5,8,10}; //from 1
//int ninthGradeDueDates[5] = {8,10,1,3,5}; //in c

//int tenthGradeMonths[5] = {1,3,5,8,10}; //from 1
//int tenthGradeDueDates[5] = {8,10,1,3,5}; //in c

int eleventhGradeMonths[7] = {1,3,5,7,8,10,11};
int eleventhGradeDueDates[7] = {8,10,11,1,3,5,7};

int twelfthGradeMonths[5] = {1,3,5,8,10};
int twelfthGradeDueDates[5] = {8,10,1,3,5};
```

Fig 3.20

Developer Manual

Version 1.1

3.4 Web pages

3.4.1 Modify existing links

All of the existing links can be modified inside one if/else statement in the “viewDidLoad” method inside “WebPageViewController.m” (Fig 3.21). The comments to the right of each line of code describe what URL each page is associated to. For an example, if the URL for the “Greetings” button is to be changed, change the highlighted text in Fig 3.21.

```
if([page isEqualToString:@"Apply"]) //url for Apply
    urlString = @"http://www.tccd.edu/Admissions/Apply.html";
else
{
    //this block gives the screen a home button in the top right hand corner
    UIBarButtonItem *homeButton = [[UIBarButtonItem alloc]
        initWithBarButtonSystemItem:UIBarButtonSystemItemAction
        target:self
        action:@selector(homeButtonPressed)];
    [homeButton setTitle:@"home"];
    [[self navigationItem] setRightBarButtonItem:homeButton];

    if([page isEqualToString:@"Greetings"])
        urlString = @"http://www.youtube.com/playlist?list=PLMi1RxsDDWS_j5-c9NktWuLQ4YA7me1pa&feature=g-list"; //url for Greetings
    else if([page isEqualToString:@"How To's"])
        urlString = @"http://www.youtube.com/playlist?list=PLMi1RxsDDWS8-ET7sa_IQ9uALw8JgnLZj&feature=g-list"; //url for How To's
    else if([page isEqualToString:@"Contacting TCC"])
        urlString = @"http://www.youtube.com/playlist?list=PLMi1RxsDDWS-QguIzmggPC8buAdk7Asg&feature=g-list"; //url for Contacting TCC
    else if([page isEqualToString:@"College Life"])
        urlString = @"http://www.youtube.com/playlist?list=PLMi1RxsDDWS82Qq_s426Y4_MIUjKyxWTb&feature=g-list"; //url for College Life
    else if([page isEqualToString:@"Career Coach"])
        urlString = @"http://tccd.emsicareercoach.com/#Radius=50&Zip=76102&AreaIDs=&action=loadHomePage"; //url for Career Coach
    else if([page isEqualToString:@"GenTx"])
        urlString = @"http://gentx.org/get-inspired/learn-more-earn-more/"; //url for GenTx
    else if([page isEqualToString:@"FAFSA"])
        urlString = @"http://www.fafsa.ed.gov/"; //url for FAFSA
    else if([page isEqualToString:@"Collegeboard"])
        urlString = @"http://www.collegeboard.com/"; //url for Collegeboard
    else if([page isEqualToString:@"Scholarship Information"])
        urlString = @"http://www.tccd.edu/scholarships/"; //url for Scholarship Information
    else if([page isEqualToString:@"ACCUPLACER"])
        urlString = @"http://www.accuplacer-test.com/"; //url for ACCUPLACER
    else if([page isEqualToString:@"Vaccine Information"])
        urlString = @"http://www.tccd.edu/MCV4/"; //url for Vaccine Information
}
```

Fig 3.21

Developer Manual

Version 1.1

3.4.2 Add Additional Links

The example below shows you how to add a link to the TCC homepage on the Career Help screen.

3.4.2.1 Create a button

To the bottom right of the screen, there will be an “Object Library” tab. Scroll through this library until the “Round Rect Button” appears (Fig 3.22). Left click and drag this object to the Career Help screen. Double click on the button to change its text (Fig 3.23).

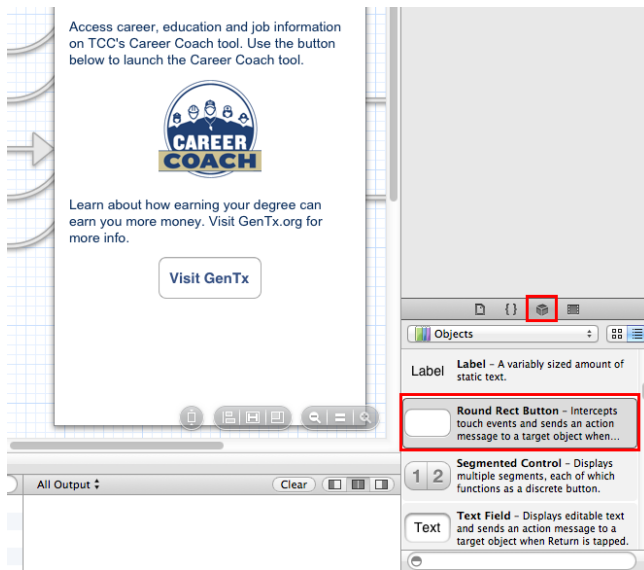


Fig 3.22

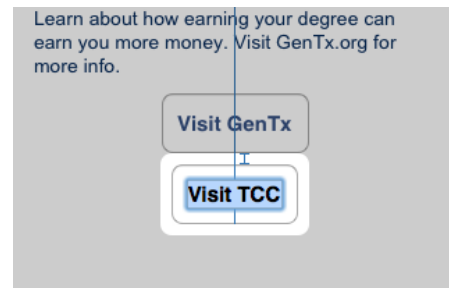


Fig 3.23

3.4.2.2 Create the segue

Right click on the button described above and drag to the WebPageViewController page in the interface builder (Fig 3.24). Release the right mouse button, and select “push” from the options presented. Select the segue that was just created. If you are having trouble identifying it, click on all segues out of Career Help; the one you just created will have your button highlighted (Fig 3.25). Now, in the 4th tab of the editor to the right, assign the segue a unique identifier (Fig 3.26).

Developer Manual

Version 1.1

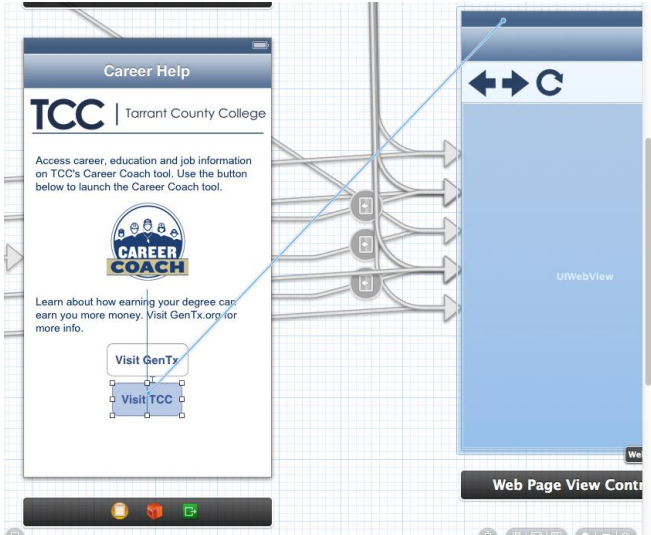


Fig 3.24

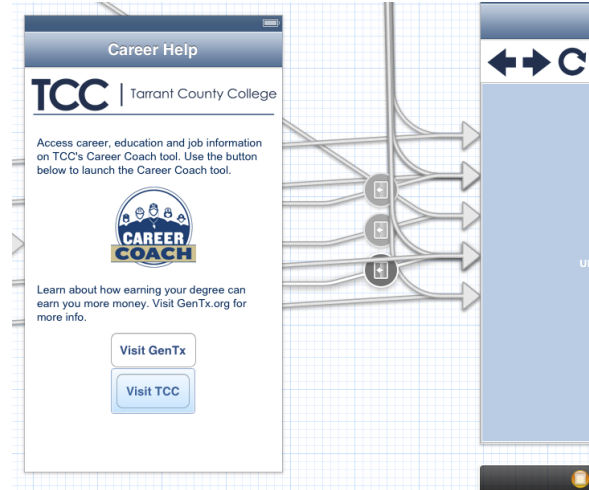


Fig 3.25

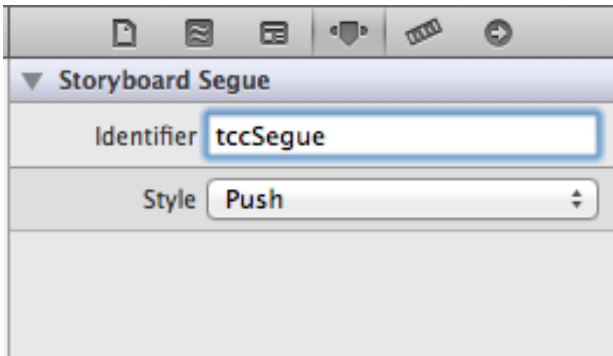


Fig 3.26

3.4.2.3 Prepare for segue

Inside the "prepareForSegue" method in "CareerHelpViewController.m", a new if statement must be added for the new segue, highlighted in Fig 3.27. Instead of "tccSegue", use the unique identifier created in 3.4.2.2, and instead of "TCC Home", use the title desired for the web page.

```
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender
{
    WebPageViewController *controller = (WebPageViewController*)segue.
    if ([segue.identifier isEqualToString:@"careerCoachSegue"])
        [controller setPage:@"Career Coach"];
    if ([segue.identifier isEqualToString:@"genTxSegue"])
        [controller setPage:@"GenTx"];
    if ([segue.identifier isEqualToString:@"tccSegue"])
        [controller setPage:@"TCC Home"];
}
```

Fig 3.27

Developer Manual

Version 1.1

3.4.2.4 Target the segue

Now, the “WebPageViewController.m” class must be set up to handle the new segue. At the end of the if/else statement shown in Fig 3.21, create a new else/if statement for the new segue, shown in Fig 3.28. Instead of “TCC Home” use the title defined in 3.4.2.3, and instead of “http://www.tccd.edu/”, insert the URL of the desired page.

```
else if([page isEqualToString:@"Vaccine Information"])
    urlString = @"http://www.tccd.edu/MCV4/"; //url for Vaccine Information

else if([page isEqualToString:@"TCC Home"])
    urlString = @"http://www.tccd.edu/"; //url for new web page
```

Fig 3.28

4 Android Application

4.1 Development Environment

The Android application was developed using the Eclipse 4.2 Juno IDE. Before development can begin, the Android SDK must be downloaded. It is recommended that you set up the ADT bundle for Eclipse. This will automate most of the SDK download and installation as well as install the appropriate version of Eclipse. There is a guide at <http://developer.android.com/sdk/installing/bundle.html> that was very helpful for setting this up.

4.2 Class structure

4.2.1 Naming System

All of the classes that are the context of a layout have abbreviated names. They may seem obscure at first, but they are the same labels used for the wire frame. A brief reference will be given (Fig 4.1) to clarify the names.

Developer Manual

Version 1.1

- A1 – Agreement page
- C1 – Career page
- CL1 – Checklist page
- CL2 – 9th Grade page
- CL3 – 10th Grade page
- CL4 – 11th Grade page
- CL5 – 12th Grade page
- CP1 – Change password page
- FP1 – Forgot password page
- HS1 – Logged out home screen
- HS2 – Logged in home screen
- O1 – Options page
- P1 – Create profile page
- SQ1 – Edit security question page
- V1 – Videos page

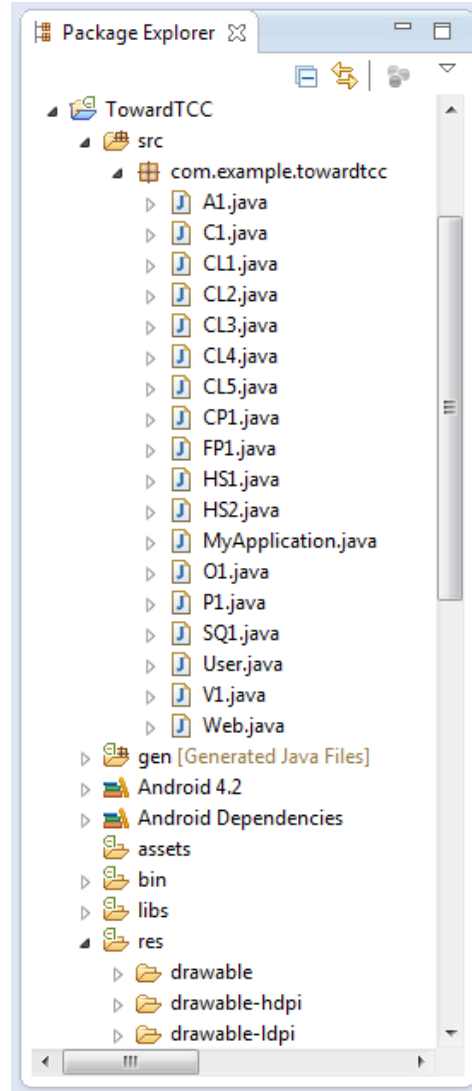


Fig 4.1

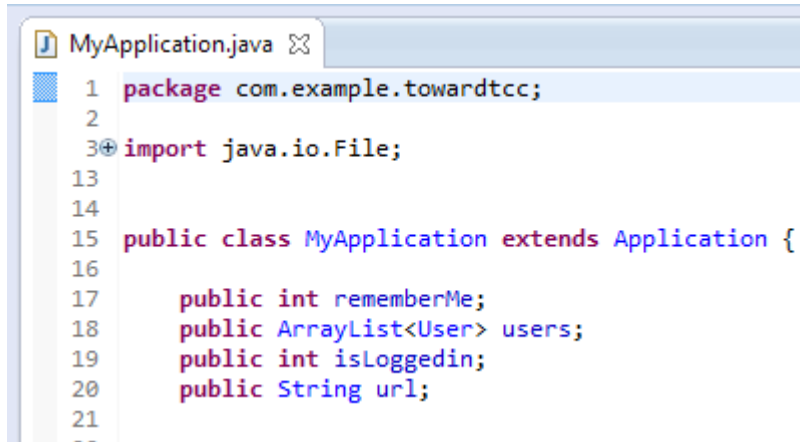
4.2.2 Important Classes

4.2.2.1 MyApplication

This class is very important. It is where all the IO takes place. This class is set to be the applications main class. This makes it possible to have global variables (Fig 4.2). The “rememberMe” variable stores the remember me check box state on HS1, the “users” array stores all the user profiles on the phone, “isLoggedin” stores what user is currently logged into the application, and “url” stores the web address that the Web class uses. Without it, none of the pages could easily pass information to one another.

Developer Manual

Version 1.1



```
MyApplication.java ✕
1 package com.example.towardtcc;
2
3 import java.io.File;
13
14
15 public class MyApplication extends Application {
16
17     public int rememberMe;
18     public ArrayList<User> users;
19     public int isLoggedIn;
20     public String url;
21
--
```

Fig 4.2

4.2.2.2 Web

The Web class is simple to understand. It will display whatever web address is loaded into MyApp.url before it is called to the screen (Fig 4.3). Therefore you must set MyApp.url to the appropriate web address before calling the Web class.

Developer Manual

Version 1.1

```
Web.java
2
3 package com.example.towardtcc;
4
5+ import android.net.ConnectivityManager;
18
19 public class Web extends Activity {
20     private WebView mWebView;
21     MyApplication MyApp;
22- @SuppressWarnings({ "SetJavaScriptEnabled", "NewApi" })
23     @Override
24     protected void onCreate(Bundle savedInstanceState) {
25         super.onCreate(savedInstanceState);
26         setContentView(R.layout.activity_web);
27         MyApp= ((MyApplication) this.getApplication());
28
29         //creates webview and sets all the nice options
30         mWebView = (WebView)findViewById(R.id.webView);
31         mWebView.getSettings().setJavaScriptEnabled(true);
32         mWebView.getSettings().setSupportZoom(true);
33         mWebView.getSettings().setUseWideViewPort(true);
34         mWebView.getSettings().setBuiltInZoomControls(true);
35         mWebView.getSettings().setDefaultZoom(WebSettings.ZoomDensity.FAR);
36         mWebView.loadUrl(MyApp.url);
37         mWebView.setWebViewClient(new MyWebViewClient());
38
```

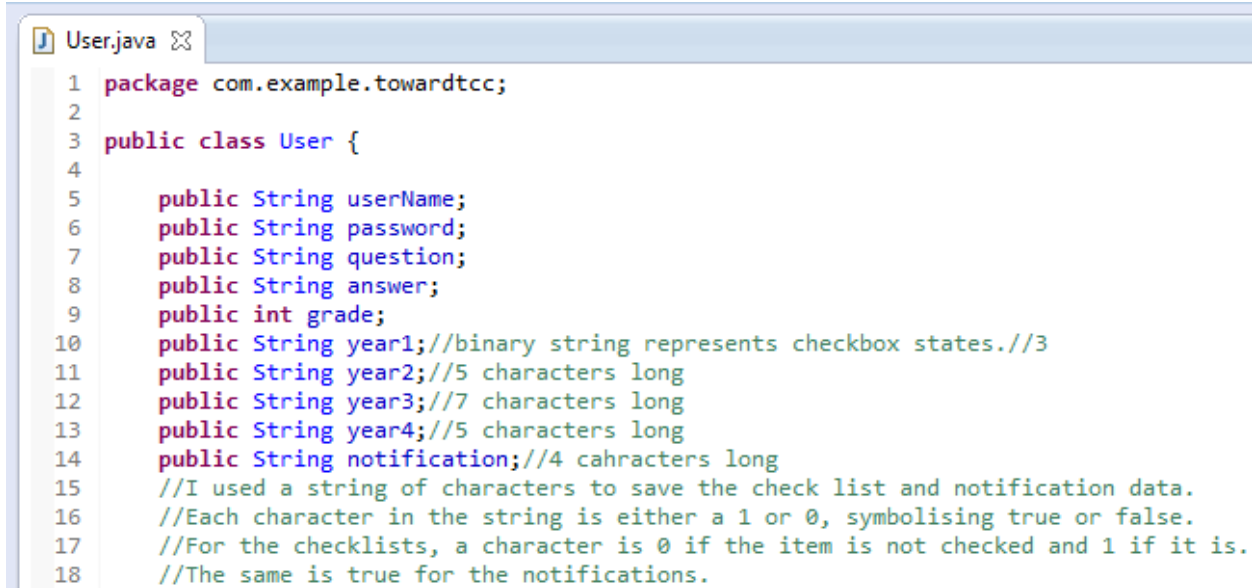
Fig 4.3

4.2.3 Objects

The only custom object in the Android application is the User() (Fig 4.4). A user consists of a username and password, along with a security question and answer. An int represents their grade. The year1 string is the data from the 9th grade checklist, year2 is the data from the 10th grade checklist, and so on. The notification string stores the state of the notification checkboxes in O1. There are no methods and only two constructors. One is used when the user creates a new profile and the other is used when loading in a profile from the device.

Developer Manual

Version 1.1



```
1 package com.example.towardtcc;
2
3 public class User {
4
5     public String userName;
6     public String password;
7     public String question;
8     public String answer;
9     public int grade;
10    public String year1;//binary string represents checkbox states.//3
11    public String year2;//5 characters long
12    public String year3;//7 characters long
13    public String year4;//5 characters long
14    public String notification;//4 cahracters long
15    //I used a string of characters to save the check list and notification data.
16    //Each character in the string is either a 1 or 0, symbolising true or false.
17    //For the checklists, a character is 0 if the item is not checked and 1 if it is.
18    //The same is true for the notifications.
```

Fig 4.4

4.3 Checklist

4.3.1 Modify Checklist text

Changing the text on the android platform is very straight forward. All of the Strings that are used in the XML layouts are stored in res/values/strings.xml. The names of the strings are formatted so that the characters before the underscore are the class they are used in, and after the underscore there is a descriptive name for the string. If small changes to the dates or wording need to be made then you only need to modify this file.

4.3.2 Add Checklist tasks

If you need to add an entirely new item to the checklist then the appropriate layout file must be modified. The layouts for all the screens are stored in res/layout/ (Fig 4.5). Select the checklist layout that needs to be modified and you will see the GUI builder. From there, it is easy to add the item to either the bottom of the list or insert it in the middle.

Now that the item is being displayed in the layout, you need to add the CheckBox ID to the corresponding class. This can be done easily by following the comments and adding the CheckBox object to the onCreate() and save() methods.

The last thing that must be done is to add the extra checklist item to the User class. Open User.java and add a "0" in the constructor method to the checklist that you modified earlier. This is essentially adding the item

Developer Manual

Version 1.1

to the user's profile. Now open the MyApplication.java file and modify the default String in the loadArray() method.

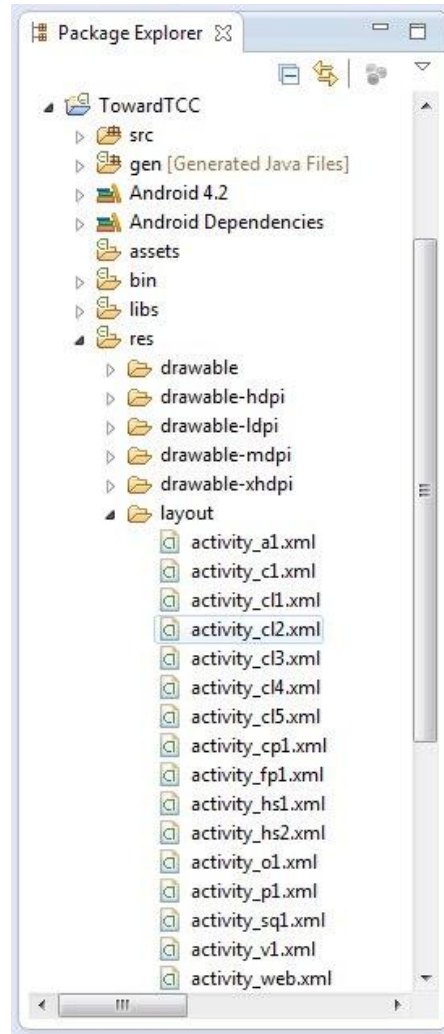


Fig 4.5

4.3.3 Modify Checklist dates

Adding additional dates to the checklist for reminders is actually a simple task. Open the HS2.java file and in the onCreate() method, add the integer month to the ArrayList. Make sure that the index of the added date is in the corresponding index of the checklist item. There are also comments explaining the layout and system used for reminders.

Developer Manual

Version 1.1

4.4 Web pages

4.4.1 Modify existing links

4.4.1.1 Checklist links

Links for the checklist can be found in the CL4 and CL5 classes. In order to change the website set the Myapp.url String to the desired web address before starting the Web activity.

4.4.1.2 Video links

Links for videos can be found in the V1 class. In order to change the website set the Myapp.url String to the desired web address before starting the Web activity.

4.4.1.3 Career links

Links for the career page can be found in the C1 class. In order to change the website set the Myapp.url String to the desired web address before starting the Web activity.

4.4.1.4 Apply Now link

The link for the Apply Now button can be found in the HS1 and HS2 classes. In order to change the website set the Myapp.url String to the desired web address before starting the Web activity.

4.4.2 Add Additional Links

Adding an additional link is simple. Create a button and set it's "on click" attribute to a method that you created. Simply set MyApp.url to the desired web address then create an Intent to start the Web activity.

Then modify the onBackPressed() method in the Web class so that it directs the user back to the page they were last at based on the string in MyApp.url.

Developer Manual

Version 1.1

5 Glossary of Terms

CACO (College Awareness Community Outreach) – They are the customers for the project. Goes to area schools to teach children how to be successful in school and how higher education can make a difference in their life.

Career Coach – Students can input their interests and receive a list of relevant careers and all the classes required.

iOS – Apple's mobile operating system.

TCC (Tarrant County College)

TCU (Texas Christian University)