DEV104 Design and Develop an iOS App

Final Project

Assessment type: App

Time/word limit: n/a

Due Date: Monday of Week 17, 10:00AM (AEST)

Weighting: 100%

Overview

This Final Project requires you to prototype and design an app. You are to choose one of the following two options:

1. Your own app idea which meets the requirements of the BYO App Project Checklist and has been approved by your mentor, or
2. A weather app that displays the current weather conditions, 24-hour forecast and 5-day forecast for 20 locations of your choice.

The purpose of this assessment is for you to demonstrate that you can follow the App Development Process – Brainstorm, Plan, Prototype and Evaluate – and then code your app after working through the process.

You'll be coding the app in the iOS Swift language to run and compile in the Xcode environment. The completed Final Project will be delivered in a zipped app project folder along with evidence of documents and assets related to the work you've completed through the App Development Process.

To ensure the best chance of success in this Final Project, you should have completed each week's Milestone activities to form the components required in the Final Project. You should have also completed the Guided Projects and the app projects within each App Build Week so you have the foundation knowledge in place to demonstrate competency in each of the assessment criteria below.

Assessment criteria

This assessment will measure your ability to:

- apply the Brainstorm and Planning tasks from the App Development Process (1pt)
- apply the Prototyping and Evaluation tasks from the App Development Process (1pt)
- build simple workflows and navigation hierarchies (1pt)
- produce error-free code that compiles and runs in the Xcode environment (1pt)
- pull data from a web API and decode into JSON (1pt)
- create an input screen to address all user requirements (1pt)
- define views and view controllers and use the Model-View-Controller pattern (1pt)
- display data in lists using UITableViews and use Auto Layout (1pt)
- persist app information across app launches (1pt)
- code programmatic UI animation (1pt)
- document code to explain app features (1pt).

Please note that in order to pass this assessment, you need to demonstrate competency in every one of the above criteria. You will need to meet all these expectations to pass the course and gain your credential. See the rubric at the end of this document for details about these expectations.

Course learning outcomes

This assessment is relevant to the following course learning outcomes:

- CLO1: Produce error free code in the iOS Swift language.
- CLO2: Create an app using the Xcode environment.
- CLO3: Create an app that uses Model-View-Controller pattern.
- CLO4: Use AutoLayout for consistency in an app.
- CLO5: Design and prototype an app to user specifications.
- CLO6: Document code to explain app features.
- CLO7: Create an app that incorporates logic.
- CLO8: Create an app that allows data to be input, updated and stored between user sessions.

Assessment Details

Create and deliver an app which has gone through the App Development Process. Then submit the app and evidence of work you have completed in the App Development Process for mentor review.

Your Final Project will consist of two parts:

Part A: Brainstorming, planning, prototyping and evaluation of your app idea.

Part B: Coding your app.

Some of the Milestones you have worked on over the course will form components of your Final Project. Ensure you have completed all of these and then follow the instructions below to complete your project:

PART A: Brainstorming, planning, prototyping and evaluation of your app idea

Before you code your app, you need to work through the App Development Process:

1. **Brainstorm and Plan**: In Milestone 5 you created a Final Project Proposal consisting of a problem statement, persona and minimum feature set. Review this document and make any adjustments if you need to.
2. **Plan:** In Activity 7.3.3, you created simple workflows for your Final Project. Review your workflows and make any adjustments if you need to.
3. **Prototype and evaluate**: In Milestone 9, you created a prototype and evaluated user feedback before making some improvements to your app design. Review these assets and make any adjustments if you need to.

Once you have Part A documents and assets in order, move on to Part B.

PART B: Coding your app

Using your documents and assets from Part A as reference, you're going to commence coding your app. For Part B, you can take one of the following options:

- If you have chosen to bring your own app idea and code it for this Final Project, ensure that it meets the requirements of the BYO App Project Checklist and has the approval of your mentor. You can then commence coding according to your plan and prototype.

- If you have chosen to complete the weather app provided for this Final Project, follow the instructions below in **Weather App**.

Weather App

Only follow these instructions if you're going to use the provided app idea for the Final Project:

1. **Set up your access to the Weatherbit.io API**
   To complete the weather app project, you'll use the Weatherbit.io API. For details and restrictions of the Weatherbit API, please refer to the "Pricing" page at https://www.weatherbit.io/pricing#free_tier.

   You will not need to provide any payment information to access the "free" tier. Upon signing up, you will be offered a trial of a paid tier — make sure to decline the offer.

2. **Test that your access to the API works**
   Beneath the surface of macOS is an entire world that you can access only from the command line. The Terminal app (in your /Applications/Utilities folder) allows you to control your Mac using a command prompt. Terminal can be used for all sorts of different tasks, and there are advantages to using it. Some of them can be performed in the Finder but are quicker in Terminal. Others access deep-rooted parts of macOS that aren't accessible from the Finder without specialist applications.

   Using a command called curl, you can download data from any URL and have it immediately printed to your screen. Open Terminal and run the following command (substituting {YOUR_API_KEY} with your actual API key, the following example will fetch the forecast for Melbourne):

   ```
   curl "https://api.weatherbit.io/v2.0/forecast/daily?
   key={YOUR_API_KEY}&lat=-37.840935&lon=144.946457"
   ```

Tips on using the Weatherbit.io API:

- Be sure to account for optional properties coming back from the API — there will be times when a forecast doesn't have one or two properties. The properties that may be returned by the API are described here: https://www.weatherbit.io/api/weather-forecast-16-day
- Exclude the weather data that you do not want displayed in your app.
- As Weatherbit.io does not provide JSON payloads in a human-readable (also called "pretty-printed") way, you may want to use a tool to tidy up the JSON API response. These tools allow you to paste your JSON API response, which it then formats in a more human-readable way. One such tool is available at: https://jsonformatter.curiousconcept.com/
- Weatherbit.io provides the images for their weather icon codes on one of their webpages (and not via another API). Consider saving these images and including them your app's assets so that you can use them in your code. The images can be found at: https://www.weatherbit.io/api/codes
- You can find documentation for all of the APIs made available by Weatherbit.io at: https://www.weatherbit.io/api
  - These will be useful, for if you want to create an even more useful iOS application, or if you want to practise reading and interpreting documentation — something that software developers do very often!

Build the weather app

Once you have set up your Weatherbit.io API access and ensured that it works, you need to build your iOS app to use the API and display the data that you can now access. Make sure that you build the app while referencing the documents and assets from Part A.

Each Forecast screen must include the information listed below for each day shown. Some of this information must be retrieved from the Weatherbit.io API.

- the date
- current temperature (current Forecast scenario only)
- weather description (e.g. Sunny/Windy/Cloudy/Rainy etc.)
- high temperature (not required in the current Forecast scenario)
- low temperature (not required in the current Forecast scenario)
- wind direction/bearing
- wind speed
- an image representing the forecast
- a favourite button that animates when users tap it to give a visual reward. The app must remember the user's favourite forecast (location and type) and load this screen on launch if available.

Technical specifications

The following specifications apply to both the weather app project and the mentor approved BYO app project:

- Create your app so it compiles and runs in the Xcode environment.
- The app code must demonstrate best practice in quality code. For example:
    - code is reused where possible to keep duplication to a minimum
    - code is split across multiple files and classes
    - code is formatted neatly and commented where necessary.
- Swift architectural patterns are used correctly (i.e. MVC, delegates, closures, protocols).
- Data screens must be designed using a storyboard in Xcode.
- The app must use the MVC architecture.
- The app must use Auto Layout.
- View Controllers in the storyboard use subviews to assist in layout.
- Bespoke structs or classes are defined as models to contain data and/or encapsulate logic.
- Structs and Classes are defined in separate files.
- API responses are decoded using the Decodable protocol.
- Files are organised logically and in groups within Xcode's navigator.
- Some user data is persisted between app launches.
- Some programmatic animations are included where it makes sense to enhance the user experience.

Audience

The app is to be produced for the user/s you identified in your customer persona in your Final Project Proposal.

Presentation format

Written documents can be in common readable formats such as Word or PDF. Any diagrams can be in any common readable formats such as scanned PDF pencil sketches, PowerPoint, JPEG or PNG files.

Remember to present your whole project folder, and not just the project file.

Resources/Hints

Your main source of information should be the work you have completed throughout the course. Other sources are websites and tutorials.

Here are some other helpful hints for completing your Final Project:

- The work you did in the App Development process is the blueprint for coding your app. Ensure you have a clear idea of the layout, content and the text that will appear before you start coding.
- Consider what model objects you'll need to support your app.
- Implement one view at a time, testing frequently.
- Break down the implementation of complex views into steps, so they only change a little bit before testing again.
- As you progress through your coding/build, regularly test your app on a friend or family member to determine whether it is intuitive and user-friendly.
- Remember to clearly document your code so that any other app developer could pick it up and understand what you have done.
- There are many free public APIs available on a wide range of topics. Have a search for something that might interest you or check out collections such as: https://github.com/public-apis/public-apis
- Keep it simple.

What feedback can you expect?

Your mentor will look at the criteria in the rubric at the end of this brief to give you feedback on your Final Project. They will be checking that your app:

- is created according to the instructions in this brief
- is created according to your plan, workflows, navigation hierarchies and prototype you created in your App Development Process
- is coded without errors in the iOS Swift language
- compiles and runs without errors in the Xcode environment
- pulls data from a web API and decodes it into JSON
- has defined views and view controllers, and uses the Model-View-Controller pattern
- displays data in lists using UTableViews and uses Auto Layout
- contains persistant app information across app launches
- contains coded UI animation
- has documented code that explains the app’s features so any other coder could pick up your code and work with it.

Submission format

Your supporting evidence of going through the App Development Process needs to be submitted along with a zip of your app project folder. Supporting evidence includes the:

- Final Project Proposal consisting of your problem statement, user persona and minimum feature set
- app workflow and navigation hierarchy
- prototype with user test results, feedback evaluation and iteration report.

When you are finished, zip your project folder and submit it through the assessment submission area.

Note: You need to upload the whole project folder, and not just the project file. Your zipped content can't be greater than 100 MB. If you're having problems with your app size, check any images you're using (use .jpg instead of .png).

Referencing guidelines

If you source any material from a primary or secondary source, please include the information in your submission. You must acknowledge all the sources of information you have used in your assessments and use RMIT Harvard referencing style for referencing.

Refer to the RMIT Easy Cite referencing tool to see examples and tips on how to reference in the appropriated style. You can also refer to the library referencing page for more tools such as EndNote, referencing tutorials and referencing guides for printing.

Academic integrity and plagiarism

Academic integrity is about honest presentation of your academic work. It means acknowledging the work of others while developing your own insights, knowledge and ideas.

You should take extreme care that you have:

- Acknowledged words, data, diagrams, models, frameworks and/or ideas of others you have quoted (i.e. directly copied), summarised, paraphrased, discussed or mentioned in your assessment through the appropriate referencing methods.
- Provided a reference list of the publication details so your reader can locate the source if necessary. This includes material taken from Internet sites.

If you do not acknowledge the sources of your material, you may be accused of plagiarism because you have passed off the work and ideas of another person without appropriate referencing, as if they were your own.

RMIT University treats plagiarism as a very serious offence constituting misconduct.

Plagiarism covers a variety of inappropriate behaviours, including:

- failure to properly document a source
- using copyright material from the internet or databases
- collusion between students.

For further information on our policies and procedures, please refer to the University website.

Assessment declaration

When you submit work electronically, you agree to the assessment declaration.

Criteria	Ratings	
	Meets expectations	**Does not meet expectations**
Criteria 1 Apply tasks of Brainstorm and Plan from the App Development Process.	A Final Project Proposal has been submitted which includes: • A definition of the problem the app is solving. • A clear vision statement for the app. • A user persona/s which identifies the type of user or users the app is aimed at. • A minimum feature set	A Final Project Proposal has not been submitted. The Final Project Proposal does not include any of the following: • A definition of the problem the app is solving. • A clear vision statement for the app. • A user persona/s which identifies the type of user or users the app is aimed
	1 pts	**0 pts**
Criteria 2 Apply tasks of Prototype and Evaluate from the App Development Process.	There is evidence of prototyping and evaluation of the app solution including: • Wireframe prototype based upon the app to be designed. • Record of usability testing feedback including assumptions/scenarios tested and commentary on user success or failure. • Record of evaluation and	No evidence of prototyping and evaluation of the app solution has been submitted. • The wireframe prototype bears no resemblance to the completed app. • No suitable records of usability testing feedback which shows assumptions/scenarios tested and commentary on user performance.
	1 pts	**0 pts**
Criteria 3 Build simple workflows and navigation hierarchies.	A workflow and navigation hierarchy shows how the app will flow and function. The app behaves as dictated by the submitted workflow and navigation	The workflow and navigation hierarchy is confusing or does not show how the app will flow and/or function. The submitted workflow and
	1 pts	**0 pts**
Criteria 4 Produce error-free code that compiles and runs.	App is coded in Swift and contains no errors. App runs in the Xcode environment as expected with no errors.	Swift code contains errors. App does not run/compile in the Xcode environment and/or contains errors.
	1 pts	**0 pts**
Criteria 5 Pull data from a web API and decode into JSON.	The app obtains data from an online service or API (such as Accu Weather). The data from the online service or API is decoded into JSON.	The app does not obtain data from an online service or API. There is no evidence that JSON is used to decode the online service or API data.
	1 pts	**0 pts**

RMIT Online

Criteria	Ratings	
	Meets expectations	**Does not meet expectations**
Criteria 6 Create an input screen to address all user requirements.	The input screen allows input of data and provides feedback which is expected and/or useful for the user. The user's taps and selections guide which screens and information are shown	Data cannot be added to the input screen. Feedback from the input screen is unavailable or does not make sense. The app is unresponsive to the user's taps and selections.
	1 pts	**0 pts**
Criteria 7 Define views and view controllers and use the Model-View-Controller	Views and view controllers are defined. There is evidence of the Model-View-Controller pattern being used.	Views and view controllers are undefined. There is no evidence that the Model-View-Controller pattern has been
	1 pts	**0 pts**
Criteria 8 Display data in lists using UITableViews and use Auto Layout.	The app contains data displayed in a list which uses UITableViews and Auto Layout.	There is no evidence of data displayed in a list which uses both UITableViews and Auto Layout.
	1 pts	**0 pts**
Criteria 9 Persist app information across app launches.	The app has the ability to save and load app data which persists across multiple sessions. A user can pick up from where they left off each time the app is used.	The app cannot save and load persistent data across multiple sessions. The user has to restart from the beginning each time the app is
	1 pts	**0 pts**
Criteria 10 Code programmatic UI animation	The app contains at least one coded programmatic UI animation. The animation adds personality, value and/or is relevant to the app.	There are no coded programmatic UI animations in the app. The animation has not been well-thought out and adds neither
	1 pts	**0 pts**
Criteria 11 Document code to explain app features.	The app's features have been clearly documented in the code. Another developer with no knowledge of the project could pick the app's code, understand it, and	There is no evidence of code documentation within the app. Another developer with no knowledge of the project would not be clear on what the features of the
	1 pts	**0 pts**

Total: 11 pts