App Development for Mobile Platforms (COIT20270)

(Assessment 4)

Yonradee Limsawat – 12268928

Quang Vinh Nguyen -12256421

Lecturer and tutor

Dr Azmat Ullah

MASTER OF INFORMATION TECHNOLOGY (MIT)

At the

Central Queensland University

Melbourne, Australia

August 2024

Table of Contents

Introduction

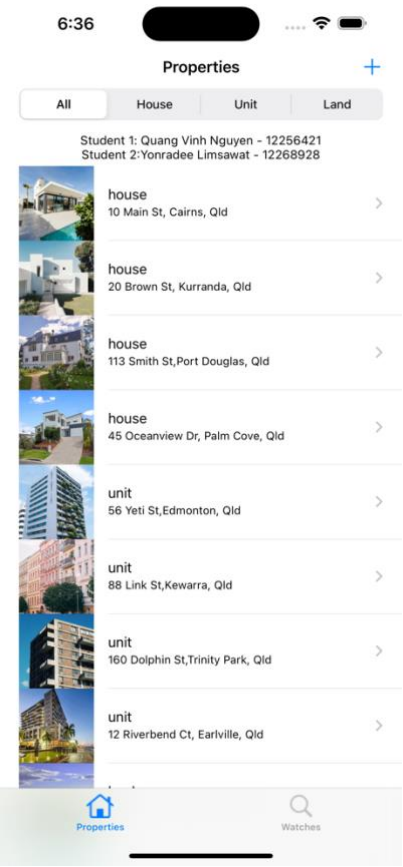
There are two major sections in this report showing the steps taken for creating, testing, and evaluating an IOS Property List application. The application provides a list of properties and has filtering features based on property type which a user can add to watchlist. The document details the hardware and software requirements, highlights the implemented features, discusses challenges encountered, and describes the testing strategy. Additionally, it includes recommendations for future improvements.

1. Hardware and Software Requirements.

The development of the Property Listing iOS application was carried out using Xcode 16.0 on a MacBook Pro M1. The application was tested on both an iPhone simulator (iPhone16) in Xcode and a physical device (iPhone 16 Plus) to ensure compatibility and performance on a real device. The minimum deployment target for the application is iOS 18.2, ensuring support for the latest iOS features and optimizations. Additionally, the app is designed to be compatible with multiple Apple devices, including iPhone, iPad, Mac, and Apple Vision, providing a versatile user experience across various platforms.

2. Application Commentary

2.1 Feature Successfully Implemented



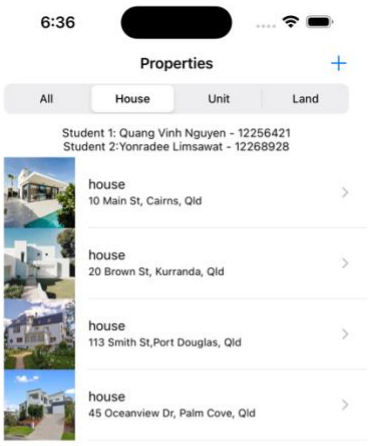
Tabs for Properties & Property Watches

The app contains two tabs which are Properties and Watches for easy navigation.

- Properties tab displays all properties.
- Watch tab shows properties added to the watch list.
- Double tap on a property adds it to the watch list.
- Single tap opens property detail's view.

Segment Controllers for Filtering

- Filters properties by type (All, House, Unit, Land).

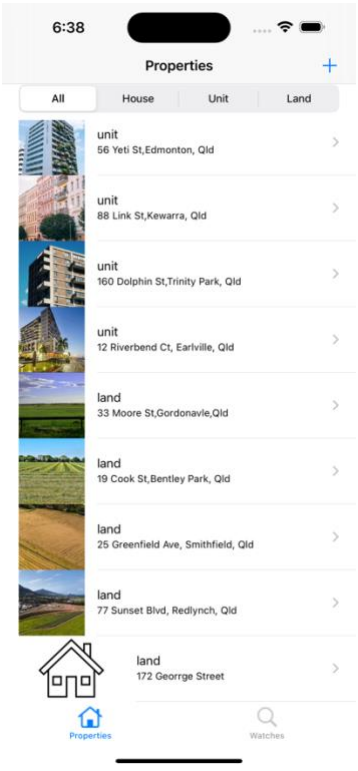


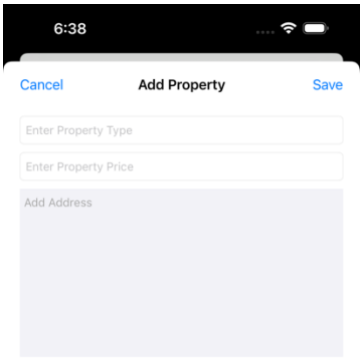
Property Data Management

- Stores property data in a JSON file.
- Retrieves data correctly to populate the property list.
- Adding New Properties

Allows users to add new properties with details:

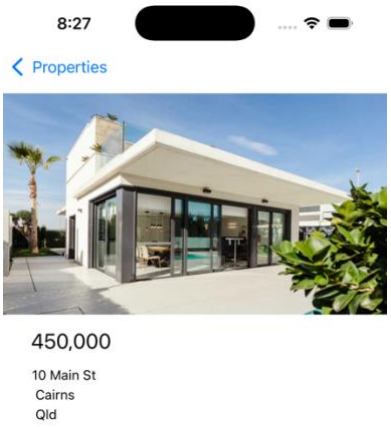
- Property type
- Price
- Address
- Uses new.jpg as default image and thumbnail.





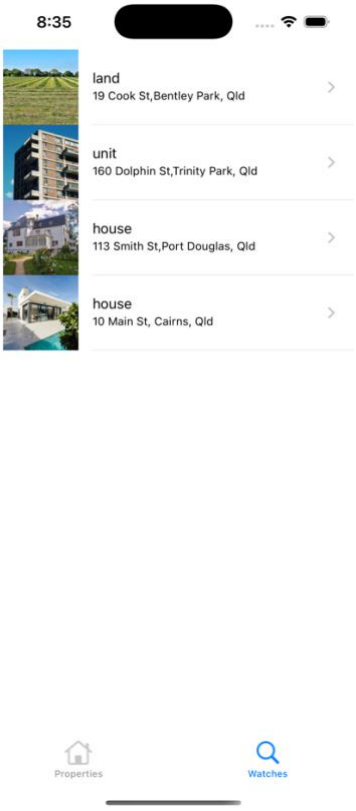
Property Details View with details:

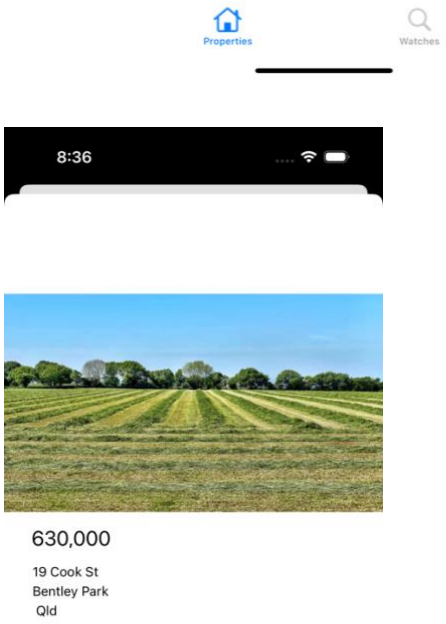
- Price
- Address



Property Watches contains:

- List of watches
- Detail of properties





2.2 Successfully Implemented Apps Functionality and Data Handling

Tab Functionalities

- Segment controllers function properly.
- Double tap adds properties to the watch list.
- Clicking a property opens its details.
- Adding new items works as expected.

JSON Data Handling

- Correctly formatted JSON file.
- Proper import and data retrieval from JSON.

2.3 Unsuccessfully Implemented Features

Some of the feature did not work as we expected despite multiple debugging attempts. Below are the key issues we encountered, and the steps taken to resolve them.

- Watchlist Not Updating After Deletion

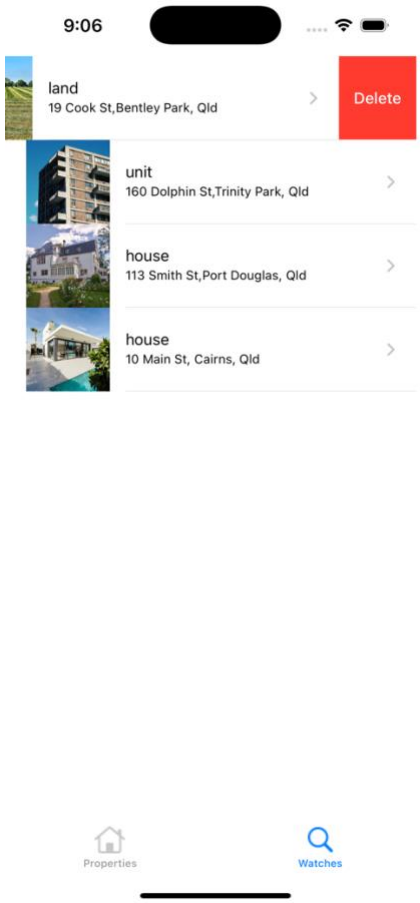
Initially, when a property was removed from the watchlist, it would reappear after adding another property. This was because the updates were not persisting correctly in UserDefaults. We attempted to fix this by making sure that the saveWatchlist() function was called every time an update occurred and confirmed that UserDefaults stored the correct data. Debugging print statements were also added to track changes in the watchlist. Although the issue was not fully resolved, the troubleshooting process helped improve the overall functionality of the app.

2.4 Additional Functionality

Swipe-to-delete on the watchlist to make this a better app. This would mean users can easily swipe to remove a property from their watchlist (from left to right, as is common practice in several modern applications).

Currently, we can only add properties to the watchlist, and there is no quick way to remove them. If the users want to clean their list, then they will need a simple and fast way to do it. Swiping was a gesture user already understood, so this functionality to remove properties could be easily integrated without the need for additional buttons or steps.

Such a feature will make the app handier as well as better the user experience. It helps to maintain the watchlist clutter-free and provides users more control over their saved homes.



3. Challenges and Fixes

During the development of my Property List app, we encountered several issues that took some time to diagnose and fix. Here's a breakdown of the problems we faced and how we tackled them:

- Group Folder vs. Reference Folder Issue

At first, we used a reference folder to organize the project files, but the code wasn't working as expected. After some troubleshooting, we realized that reference folders don't properly link files in Xcode. Switching to a group folder fixed the issue.

- JSON Parsing Problems

Initially, JSON file wasn't responding, so we added print statements to check if it was loading correctly. The file loaded fine, so we used JSONLint to validate the formatting. It turned out our JSON structure was correct, but we needed to adjust how we were reading the data in Swift.

- Table View Prototype Cells Not Showing

The table view wasn't displaying any content, so we checked if the number of rows was being detected. It turned out the problem was with the cell identifier. We had a mismatch between what was set in the storyboard and what was referenced in PropertyListViewController. Fixing the identifier resolved the issue.

- Images Not Displaying

The images weren't showing up in the app. First, we checked if they were properly added to the asset catalogue. Then, we realized that the JSON file contained full image paths, but in PropertyListViewController, we were only using the image name. Updating the JSON to match how the images were referenced in the code solved this.

- Add Property Function Not Updating UI

When we added a property, it printed that it was saved, but it didn't show up in the UI. After checking the code, we realized that we had forgotten to set the segue identifier in the storyboard to match the code. Once we fixed this, new properties appeared as expected.

These challenges helped us improve our debugging skills and gain a better understanding of Xcode, Swift, and UI development. While most issues were resolved, we are still working on fixing the remove Watch list issue.

4. Testing strategy

Both emulator and real-device testing will be conducted to make sure the Property List app functions properly and offers a seamless user experience. The approach with specific examples for every testing stage is provided below.

4.1. Target Devices

The app is designed for iOS and will be tested on iPhone 12, 13, 14, 15 and 16 series to cover various screen sizes and resolutions.

- iPhone 12, 13, 14, 15, and 16 series – Standard and Pro models for different screen sizes
- iPhone 14 Pro Max – Large screen

4.2. Testing on Simulators vs. Real Devices

4.2.1 Emulator Testing (Xcode Simulator)

Purpose: Quickly check basic functionality, UI layout, and navigation.

- Run the app on the iPhone 14 simulator to check if the property list loads correctly.
- Verify that tapping a property opens its details view without crashing.
- Use print statements and debugging tools to check if the JSON file loads data properly.

4.2.2 Real Device Testing

Purpose: Validate real-world scenarios, including touch gestures, performance, and data persistence.

- Test on an actual iPhone 14to verify double tap to add to the watchlist works smoothly.
- Check if swiping left-to-right removes properties from the watchlist (if implemented).
- Restart the app and confirm that watchlist items remain saved.

4.3. Functionality Testing

Feature	Test Scenario	Expect Result	Reality
Property list display	Open the app and check if properties are listed.	All properties from JSON should appear.	All data is loaded correctly from JSON file.
Property detail view	Tap on a property to open details.	Correct details show	Move to detail page when tap on a property.
Adding new property	Enter new property details a save.	New property shows in the list.	Click on add new property button move to new property page. Save property with sample image and details. Show new property in property list.
Filtering property by type	Use segment controllers to filter Houses, Units, and Land.	Only selected property type should be visible.	Filtering property by type and showing correctly.
Watchlist feature	Double taps a property to add it to the watchlist.	Double tap on property to add to watchlist.	Double tap to add property to watchlist and watchlist shows correct property.
Swipe to delete	Swipe a property in the watchlist to remove	Property should be deleted and stay removed after restart.	Swipe to delete property in watchlist. The removed property doesn’t show in

4.4. UI Testing

- Test in dark mode to check if text and icons are still readable.
- Rotate the device to landscape mode to see if the layout adjusts correctly.
- Try tapping buttons multiple times quickly to check for responsiveness and possible crashes.

By testing on both simulators and real devices, we can make sure the app not only functions well but also provide a smooth, good experience for the users, and handles errors well. Regular testing at each stage helps identify and fix issues before finalizing the app.

Conclusion

Developing the Property List application as a team was an enriching and collaborative experience that helped us enhance our technical skills and teamwork abilities. By assigning roles based on our abilities, we streamlined the development process and ensured that each app component was implemented efficiently.

Collaboration allowed us to identify and resolve difficulties more efficiently. When one of us met a problem, such as a JSON parsing error or a UI layout differences, the other offered a new perspective, which frequently resulted in faster solutions. This collaboration significantly improved our debugging abilities and increased our knowledge of Swift, Xcode, and iOS.

To give customers a simple, easy way to browse and filter properties while keeping track of a watchlist, the Property List application was created. Among the main features of our work were the creation of a user-friendly watchlist system, JSON data handling, and segment controllers for filtering. Several crucial features were successfully incorporated, but we also ran into problems with data durability and property filtering, which necessitated iterative testing and improvement.

Overall, this project highlights the importance of teamwork in software engineering and offered invaluable practical experience in developing mobile apps. With further refinements, such as improved filtering logic and enhanced UI responsiveness, the Property List application has the potential to become a polished and reliable tool for users interested in property listings.