Boston University
Electrical & Computer Engineering
EC464 Capstone Senior Design Project

User's Manual

Scan it! Pack it!

Submitted to

Benni Trachtenberg
btt@bu.edu

by

Team #14
Scan It! Pack It!

Team Members

Tristen Liu tristenl@bu.edu
Daniellia Sumigar dsumigar@bu.edu
Juan Carlos Vecino jvecino@bu.edu
Ramy Attie attiera@bu.edu

:
Submitted: 4/19/2024

Scan It! Pack It! – User Manual

Table of Contents

Executive Summary

Container and packaging material waste poses issues for resource management and sustainability. Shipping companies, storage industries, and ordinary people alike face the difficult task of finding optimal packaging. To address these issues, Scan It! Pack it! is a user-friendly iOS mobile application that can automatically detect object and container dimensions and generate an optimal packaging schematic. The application utilizes LiDAR sensing to perform dimension detection, cloud technology for data processing, and machine learning optimization techniques to accurately identify optimal packaging. It will encourage proper packaging methods to reduce waste and ensure the safety of goods when transported.

1 Introduction - Daniellia Sumigar

In 2018, the Environmental Protection Agency (EPA) reported 82.2 million tons of solid waste from containers and packaging in the United States. Packaging containers are necessary to protect various products, including food and medications. They are used especially in shipping and delivery services to safely transport goods from one destination to another. However, the excessive waste from containers only poses further harm to the environment.

Scan It! Pack It! addresses this problem by revolutionizing the packing experience through automatic dimension detection for a user-friendly, optimal packaging solution. Scan It! Pack It! is an iOS application designed for iPhones 12+ Pro and above that combines LiDAR sensing, cloud technology, and optimization algorithms. By leveraging the LiDAR camera, users are able to determine the dimensions of a container and objects both quickly and accurately. On the cloud, the software backend receives the dimensions and processes them into the packing algorithm which returns the packing order for the optimal packing schematic displayed to the user. To ensure a quality packing experience, Scan It! Pack It! also informs users of any items that will not fit and the space utilization for each individual container. Thus, Scan It! Pack It! provides a simple, accessible packing solution that does not require any additional hardware beyond an iOS smartphone device.

2 System Overview and Installation - Tristen Liu

Scan it! Pack it! is a software only application developed with Swift and Python for iPhones 12+ Pro and above. This application utilizes the LiDAR scanner inbuilt to these devices in order to scan objects, as well as the ARKit, UIKit and SceneKit Apple development APIs. The ARKit API allows access to the Augmented Reality functionality of iOS devices, the UIKit API is needed to implement the User Interface, and the SceneKit API facilitates easy development of 3D environments for the schematic display.

Figure 2.1 demonstrates a high level overview of the system, including what occurs on the User Side, in the backend, and what processes need to be accomplished over the cloud in AWS. The user utilizes ARKit LiDAR in order to scan points, the dimensions retrieved from LiDAR are displayed on screen using UIKit, final dimensions are sent to the cloud, and finally the schematic is shown to the user with an animation created by SceneKit.

2.1 Overview block diagram

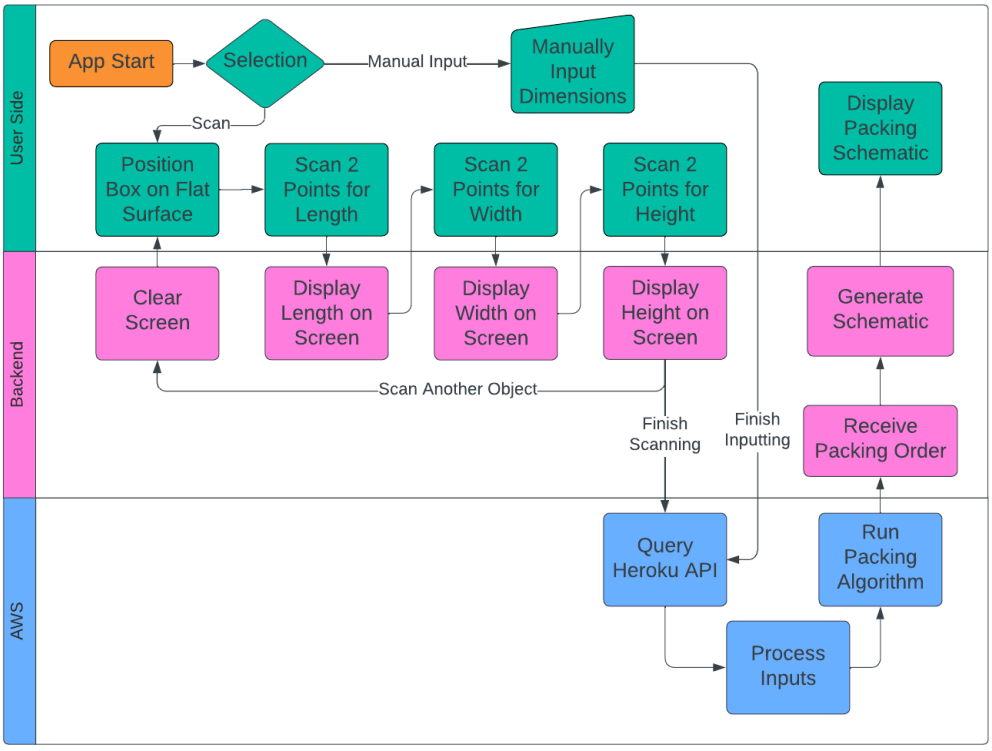

Figure 2.1 System overview of the software, detailing the User Side, Backend, as well as external AWS Cloud Computing.

2.2 User interface

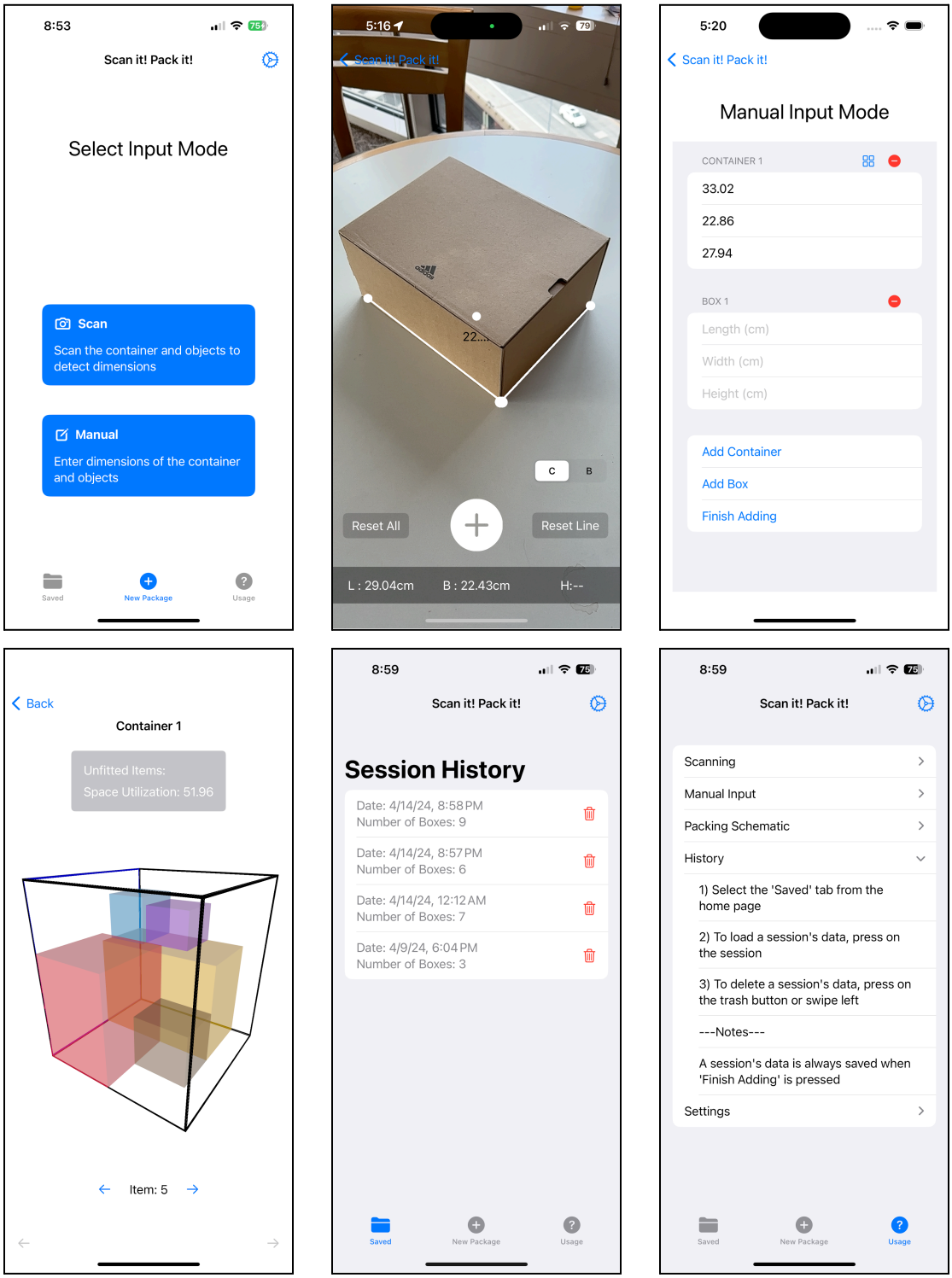

Figure 2.2 iOS Application User Interface Screens

2.3 Installation, Setup, and Support

Installation will require the user to have a MacOS Operating system with Xcode 15 iOS development framework installed, as well as an iPhone/iPad 12+ Pro device with LiDAR. Refer to Appendix 7.2 for guiding images.

1) Open Xcode and select Clone Git Repository
2) Enter "https://github.com/TristenLiu/Scan-it-Pack-it.git" as the repository URL and Clone
3) Connect the iPhone/iPad to the MacOS via USB while the Xcode project is open and wait for Xcode to connect with your device
4) Turn on Developer Mode on the iPhone/iPad in Settings > Privacy & Security > Developer Mode
    a) The device will need to be restarted in order to turn Developer mode on
5) **If the iOS development framework was not installed on Xcode installation:** At the middle of the top bar on in Xcode, click Get at the top of the window to retrieve devices (*while your device is plugged into the MacOS*) and wait for the simulator download to finish
6) In Xcode, click on the project name ScanitPackit with the blue app store icon at the top of the file list to open the app settings
7) Navigate to ScanitPackit > Signing & Capabilities
    a) Next to Status in the middle of the screen, click "Add Account" and sign into your Apple ID (*ignore this step if your Apple ID is already signed in*)
    b) In Signing & Capabilities, find the Teams dropdown and select your Apple ID as the team
    c) Change the first part (*before the period*) of the bundle identifier to a different text value
8) Select your device in the drop down list at the top of the window
    a) If your device does not appear in the drop down list, click Manage Run Destinations at the bottom of the list and connect your device
9) Build and Run the application on your device by pressing the Play button or CMD+R, and wait for the cache symbols to be copied
10) A popup will appear with the message "Unable to Verify App" on your device.
    a) On your device, go to Settings > General > VPN & Device Management > Developer App and press "Trust "Apple Development: name@example.com""
    b) Press Verify App
11) If the app does not open, Build the application again and Open ScanitPackit on your device.

3 Operation of the Project

3.1 Operating Mode 1: Normal Operation - Ramy Attie

Scanning Objects

1) Navigate to "Scan" from the home menu
2) Place the object to be scanned on a flat surface
3) Press the (+) button to capture a single point node. Keep the device steady during scanning
4) Press the (+) again on the other point on the same edge to capture the dimensions of one edge of the box
5) Repeat for each edge of the box (Length, Width and Height)
    a) When scanning the Height of an object, rotate the object such that the height is flush with the ground
6) When finished scanning, return to the home page and enter Manual mode to send the data to the server

- The "Reset All" and "Reset Line" buttons clear all measurements or the current active measurement respectively
- Use the [C | B] slider to determine whether the current scan will be saved as a Container object or a Box object
- When scanning Containers, scan the internal dimensions of the box to account for wall thickness
- Maintain a distance of at least 1ft (0.3m) away from the object for accuracy
- Keep track of the order of scanned objects for reference in the schematic

Manual Input Screen

1) Press "Add Container" or "Add Boxes" to add new fields to the screen
2) Select Container presets by pressing the 2x2 grid button on the top right of the field
3) Input object dimensions by selecting the appropriate field and entering the dimension value
4) Delete objects by pressing the (–) button on the top right of the field.
5) When finished, Press "Finish Adding" in order to view the schematic

Schematic View

1) Orient the Container with respect to the Schematic view
2) Press the left and right arrows to view different packing steps in the schematic
3) According to the Container and Box number on screen, fit the boxes into their respective containers as shown
4) Move between multiple containers using the bottom right and left arrows

- Boxes that were unable to be fit into the provided Containers will be listed at the top of the screen by number
- The initial view of the Container is top-down. The bottom of the Container is colored red

Home Screen

- Select Scanning or Manual Input views in the "New Package" middle tab
- View and Load session history in the "Saved" tab on the left-hand side.
    - Delete sessions by swiping left or clicking the trash button.
    - Load session history by pressing on one of the sessions in the list. This will lead back to the second tab, where the user can move to Scan new objects or to Manual Input
- Read the instructions in the "Usage" tab on the right-hand side
    - Expand instructions by pressing on the respective item.
- Modify settings by pressing the "Settings" button in the top right corner
    - Modify the display unit to show either Inches or Centimeters
    - Toggle prioritized packing based on necessity

3.2 Operating Mode 2: Abnormal Operation - Tristen Liu

Scanning

- If the object is too close to the device camera, the nodes may drift from their original position at the point of the object. Reset the current line and replace the points while maintaining a 1+ ft distance from the object.
- The object can be safely moved around in the middle of a scan, and the position of the line nodes can be disregarded after the scanning of an edge is completed.
- If a point or a line persists after starting a new measurement, it can be safely ignored.

Manual Input

- If a Container/Box field is added and dimensions are not inputted into the fields, the packing API will fail and a schematic will not be queried. Return to the Manual Input screen and either delete the object or fill in the required fields.

4 Technical Background

4.1 LiDAR Scanning - Tristen Liu

The Scanning portion of this application utilizes Apple's ARKit to retrieve coordinates and distances from space, and SceneKit to visualize the nodes and lines in the space.

When the user presses the (+) button, a new point node object is placed in space using SCNKit and added to a list of nodes corresponding to the current scanning state (Length → Width → Height). Additionally, a realtime line node is added to the scene to visualize the current measurement line from the camera point to the previously added point node. When the user presses the (+) button again, a second point node is created and a solid line node is added to the scene between the first and second point nodes. After two point nodes are created for the scanning state (Length → Width → Height), the scanning state will cycle to the next measurement value. If all three measurements are taken (i.e. there are six point nodes in the scene) and another point node is added, the scene will be cleared of all nodes before adding the new point node.

The distance is calculated between the two points and stored in an internal dimensions array that contains the dimensions of the current measurement. After all three measurements are taken, depending on the value of the [C | B] slider, the internal dimensions array will be appended to the corresponding global dimensions list that is shared across all of the files. This is to allow other views, such as Manual Input and Schematic View, to access the shared dimensions.

4.2 Manual Input - Tristen Liu

The Manual portion of the application is built entirely in SwiftUI. The dimensions are separated into two different categories, Containers and Boxes, which are displayed as a List of fields on the screen.

The Containers are displayed with a button that brings up a new ConainerSelectionView with a list of Container presets. Both input fields include a button to delete the specified index from the list, as well as three input TextFields that allow the user to add/modify the dimensions of the boxes. Finally, three buttons are displayed at the bottom of the screen, allowing users to Add Containers, Add Boxes or Finish Adding as necessary.

4.3 Packing Algorithm - Daniellia Sumigar

The packing algorithm is an open-source library that combines the Best Fit and First Fit Decreasing heuristics to determine the optimal placement of boxes in a given container. Starting at the bottom-left corner of the container as the origin, boxes are rotated until a suitable orientation is found. If no such orientation exists, the algorithm moves onto the next item, while keeping track of any items that could not be packed for subsequent attempts. By default, the packing algorithm will attempt to pack the largest items first. The packing algorithm also accepts a priority parameter that will prioritize placement of boxes within the container.

4.4 Packing REST API & Heroku - Daniellia Sumigar, Juan Vecino

The Packing API was built using Flask and Python and deployed on the Heroku cloud service.

The API sends a POST HTTP request, sending input information about the containers and objects, and retrieves packing information in JSON format. The output JSON keys are mapped to a SwiftUI data model for use in the schematic generation.

```
{
  "bins": [
    {
      "bin_name": "box1",
      "width": 5,
      "height": 10,
      "depth": 5
    }
  ],
  "items": [
    {
      "partno": "item1",
      "name": "Item 1",
      "typeof": "cube",
      "width": 3,
      "height": 3,
      "depth": 3,
      "weight": 2,
      "level": 1,
      "loadbear": 20,
      "updown": 0
    }
  ]
}
```

```
[
  {
    "bin": "box1(5x10x5, max_weight:100) vol(250)",
    "bin_name": "box1",
    "fitted_items": [
      {
        "dimensions": [
          "3",
          "3",
          "3"
        ],
        "partno": "item1",
        "position": [
          "0",
          "0",
          "0"
        ],
        "rotation_type": 0,
        "volume": 27.0,
        "weight": "2"
      }
    ],
    "residual_volume": 223.0,
    "space_utilization": "10.80",
    "unfitted_items": [

    ],
    "unpack_item_volume": 0,
    "used_time": 0.01255488395690918
  }
]
```

Figure 4.1 Sample Input Data (Left) and Output Data (Right) sent and retrieved by API

The Flask API on Heroku utilizes Gunicorn as its WSGI HTTP server. Gunicorn dynamically generates a server instance each time an endpoint is accessed; when there are no requests, the server remains inactive. This functionality enables Heroku to handle multiple simultaneous requests efficiently by initiating parallel server instances, thus optimizing parallel processing capabilities. Additionally, the codebase is hosted on GitHub, allowing for automatic redeployment on Heroku whenever updates are pushed to the repository. This integration streamlines updates and maintains the application's availability and functionality.

4.5 SceneKit Schematic - Daniellia Sumigar

The packing schematic was developed using Apple's SceneKit graphics framework. Upon receiving the data from the API, the resulting packing information is used to determine the optimal packing schematic. The position coordinates (X, Y, Z) determine the relative location of boxes and the rotation indicates the orientation of the boxes within the container.

5 Relevant Engineering Standards - Daniellia Sumigar, Juan Vecino

5.1 REST APIs

REST APIs define a set of protocols for application programming interfaces that conform to the representational state transfer architectural style (REST). This enables communication between computer systems through HTTP requests to access and use data.

HTTP requests define the set of actions to be done on a server. The most common HTTP requests include GET, POST, PUT, and DELETE. A GET request will retrieve data from a server, whereas a POST request will submit data to the server. PUT requests will update data and DELETE requests will delete data on the server. The data is represented in various formats including HTML, JSON, Python, PHP, and plain text.

A URI (uniform resource identifier) is sent along with the HTTP request that identifies the resource to be acted upon – an example is a URL specifying the location of the resource on the web. The HTTP request and the URI are processed in the server which then returns a HTTP response status code, indicating if the request was successful along with any requested data back to the client.

The "Scan It! Pack It!" application utilizes a REST API hosted on Heroku to perform a POST request to send and retrieve data from the Python backend.

6 Cost Breakdown - Juan Vecino

There are no software development costs for the application. The project was built using Xcode, which is free to download on a Mac Operating System.

For future iterations of this application, potential costs lie in the enrollment in the Apple Developer Program, which is required to deploy the application to Beta Testers and to the App Store.

Another cost consideration is the use of Heroku for hosting services. The cost of Heroku depends on the time the server is actively running, priced at approximately $0.010 per hour.

Project Costs for Production of Beta Version (Next Unit after Prototype)				
Item	Quantity	Description	Unit Cost	Extended Cost
1	1	Apple Developer Program – Annual Fee	$99	$99/yr
2	1	Heroku Server Hosting	~$0.010/hr	Variable
			Beta Version-Total Cost	$99 + Variable

7 Appendices

7.1 Appendix A - Specifications

Requirements	Value, Range, Tolerance, Units
Software Application	• Limited to iOS Pro Devices
Object Scanning	• Accuracy of +/– 0.1 inches under good lighting and surface conditions • Effect of user speed on scanning accuracy is negligible • Full dimension scan of an object requires 20s on average
User Experience	• Tabs and Buttons are intuitive and self-explanatory • Usage information is clearly present for the user at any time
Cloud Server/Schematic Generation	• Data processing time <1s • Generate schematic within 1s

7.2 Appendix B - Installation Images

Figure 7.2.1 Supporting image for installation step 1

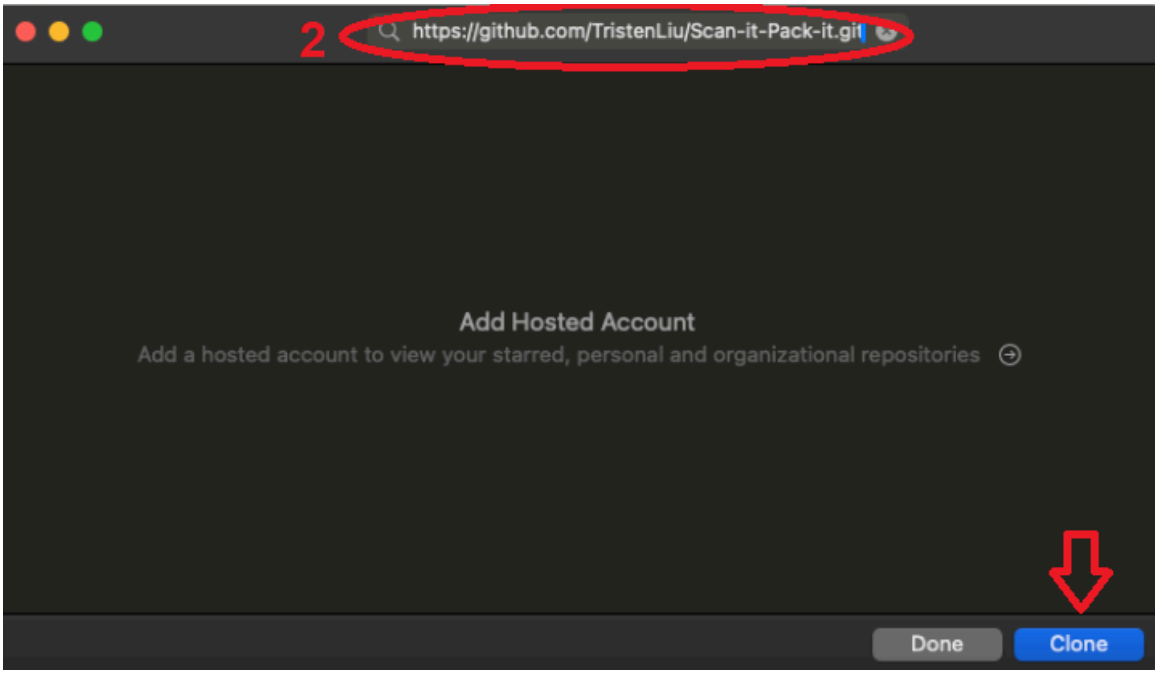

Figure 7.2.2 Supporting image for installation step 2

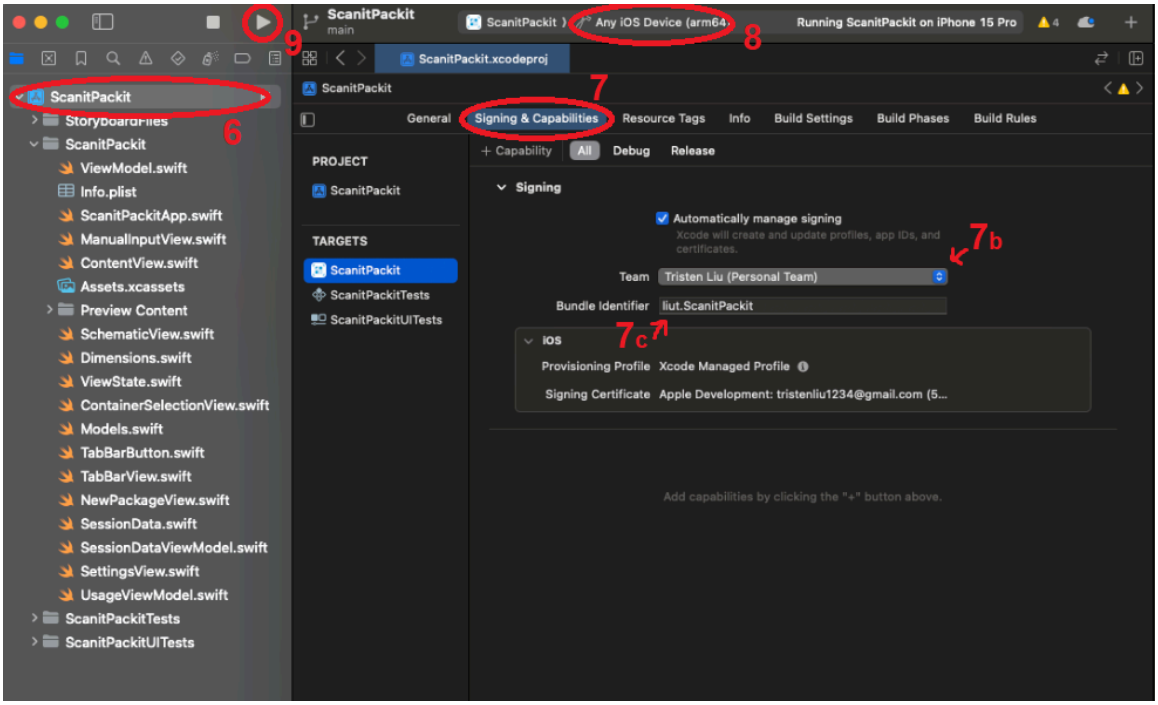

Figure 7.2.3 Supporting image for installation steps 6-9

7.3 Appendix C – Team Information

Tristen Liu (tristenl@bu.edu)
- Graduating Senior in Computer Engineering at Boston University | College of Engineering
- Worked primarily on the Scanning, Manual Input and Settings portion of the application

Daniellia Sumigar (dsumigar@bu.edu)
- Graduating Senior in Electrical Engineering with Concentration in Machine Learning at Boston University | College of Engineering
- Worked on Packing API and schematic generation

Juan Vecino (jvecino@bu.edu)
- Graduating Senior in Electrical Engineering at Boston University | College of Engineering
- Worked on Packing API and Scanning

Ramy Attie
- Graduating Senior in the ECE Department at Boston University | College of Engineering