SECURING E-COMMERCE DELIVERIES: AN INTEGRATED MOBILE APPLICATION AND PARCEL LOCKER SYSTEM FOR MITIGATING PORCH PIRACY

JonathanZiqi Chen[1], Matthew Xuehao Li[2], Jonathan Sahagun[3]

[1]Portola High School, 1001 Cadence, Irvine, CA 92618
[2]Portola High School, 1001 Cadence, Irvine, CA 92618
[3]Computer Science Department, California State Polytechnic University, Pomona, CA 91768

ABSTRACT

With the rapid surge in porch piracy, largely fueled by the proliferation of online shopping, there is an urgent need for innovative solutions to secure package deliveries [8]. This paper addresses the pressing issue of package theft through the development of a comprehensive solution that integrates a mobile application and a secure parcel locker system. Recognizing the limitations of existing methodologies, our proposed solution places a strong emphasis on technical robustness, user experience, and security.

Our approach revolves around a user-friendly mobile application catering to both consumers and couriers. This application seamlessly interfaces with a network of secure parcel lockers, creating a robust ecosystem. Leveraging Google Firebase for data storage and Android Studio Code for programming, our system ensures secure and efficient access [9].

Overcoming challenges like efficient communication with Raspberry Pi and optimal locker design layout, we utilized wireless technology and innovative design concepts [10]. Rigorous experimentation encompassing usability, effectiveness, and user satisfaction assessments has demonstrated the solution's efficacy in thwarting unauthorized access.

The results underscore the significance of a user-centered design, technical feasibility, and comprehensive security measures. By offering a multifaceted approach that prioritizes convenience, security, and user-friendliness, our solution presents a compelling remedy to the pervasive problem of porch piracy.

KEYWORDS

Android studio code, Mobile application, Flutter, Online database

1. INTRODUCTION

Porch piracy is defined as the seizure of packages outside residential housing or a business with the intent of deriving personal monetary gain. This problem has only recently begun to occur frequently in the last three decades largely in part due to the drastic expansion of online retailing during COVID-19 lockdown, causing a 31.7% increase in the number of Americans falling victim to porch piracy from May 2020 to March 2022 (41% - 54%) [4]. This is particularly problematic due to the lack of an effective deterrence in the way of preventing thieves from stealing valuable packages from people and businesses daily. In fact, a total of $2.4 billion was lost in the form of stolen goods in 2022 alone. Additionally, a "Survey data of 562 participants from 49 states reveal that nearly one quarter (23.8%) have experienced package theft" [1]. Porch piracy has far reaching implications that can significantly impact vulnerable demographics, denying individuals in dire need of medical supplies[2]. Although 44% of those who report package theft cases have sufficient evidence with a doorbell camera, only deliveries and parcels from the USPS are subject to federal punishment, leaving the majority of package theft cases to be classified as a misdemeanor and consequently enabling a significant number of thieves to avoid capture and prosecution [3]. Further, additional information on the impacts of porch piracy is unclear, with a 21% lower chance of reporting the crime if the package stolen was under $50 in value.

Methodology A

Explores consumer preferences for last-mile parcel delivery in Australia, comparing drones, postie, and parcel lockers. It assesses competitive priorities, willingness to pay, and contextual moderators. The study neglects technical and user experience aspects, and lacks insights into technical feasibility. Our project builds on this by addressing technical robustness, usability, and security, aiming for a more comprehensive solution to prevent porch piracy.

Methodology B

Introduces NFC-enabled smartphones for efficient shopping mall locker usage [11]. It streamlines deposit and retrieval processes. However, it overlooks potential security issues and doesn't consider technical integration across multiple stores. Our project improves this by integrating secure storage, user-friendly app, and robust technical aspects, enhancing security, user experience, and feasibility.

Methodology C

Evaluates sustainability of collection and delivery points (CDPs) for online orders in urban areas [12]. It examines walking distances for pick-ups and highlights a need for shared CDPs. The study doesn't consider user experience and technical feasibility comprehensively. Our project addresses these by combining secure storage, user-friendly app, and technical robustness, providing a holistic solution that caters to both sustainability and user satisfaction, with improved security measures.

Our solution is a mobile application that can be used to open a parcel locker once your package arrives in it. The account and package data for our solution is stored on Google Firebase, which is a secure database that can be used to communicate with the mobile app. Our app is used by both customers and couriers, there are two separate signups and logins that are used by each. Once the courier updates the status of the delivery, the customer can then go to the locker and unlock it using the app. The code we use to make everything run is on Android Studio Code, using a language called Dart. It will solve our problem of porch pirates stealing packages by having a secure space for packages to be stored, while also being accessible to the rightful owner of the packages. Since the locker is secured using a solenoid lock instead of a traditional lock using a key, it cannot be lock picked. We think that this is an effective solution because most porch pirates are regular people without the means to break into a locker such as ours. For example, most people don't have the skills or tools to cut through a locker door without being noticed. Our solution would be better than other methods such as security systems, because it prevents the problem while security systems can only alert the package owner. Our users also don't have to worry about their packages being stolen when they are away from home. Finally, parcels aren't exposed to the elements unlike other methods like a doorbell camera.

Experiment 1: Effectiveness and Usability Evaluation

In the first experiment, the aim was to assess the effectiveness and usability of a mobile app and secure parcel lockers in preventing porch piracy. Ten participants engaged in tasks simulating real-world scenarios, evaluating the app's functionality and locker system. Usability issues, such as initial confusion among consumers, led to suggestions for clearer instructions. Effectiveness testing showcased a consistent success rate in preventing unauthorized access. The experiment highlighted the importance of user-friendly design and the technical robustness of the solution. The results confirmed its potential to mitigate porch piracy effectively, contingent on enhancing user experience and fine-tuning technical components.

Experiment 2: User Satisfaction Assessment

The second experiment aimed to gauge user satisfaction with the same mobile app and secure parcel lockers. Participants rated their satisfaction for tasks and the overall solution on a 1-10 scale. Task-specific ratings demonstrated user satisfaction, with slight variations. Notably, task 2 received slightly lower ratings, signaling room for improvement. Qualitative feedback emphasized the app's convenience, security, and areas of customization. The findings underlined the significance of user-centered design and its impact on user satisfaction. The solution's potential in thwarting porch piracy was corroborated, contingent on addressing usability concerns and aligning with user preferences. Overall, both experiments underscored the necessity of a holistic approach to technical functionality and user experience.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. How to let the app communicate with the Raspberry Pi

One major component of our program is the page that allows the user to pick up their package from the locker. Some problems we had to consider were how to let the app communicate with the Raspberry Pi, how to set up the Raspberry Pi to control the solenoid locks, and how to update the database when statuses for the packages changed. We could use a database similar to firebase to allow the app to communicate with the Raspberry Pi. The flutter app that we built could be used to update the database online, and the database would then allow us to update the statuses of packages and the raspberry pi would be able to lock and unlock the solenoid locks accordingly based on them.

2.2. The Design

Another barrier we faced in the development of this locker is of the design itself, and how we could minimize the clutter from wiring. Some problems we had to consider were on how to hide the majority of the wiring, how the locker's design could be used to assist in organization of the components, and what we needed to solder. We could designate a single locker for the sole purpose of storing the Raspberry Pi and relay and utilize the small spaces found on the corners of each locker to slide the wiring through. We could additionally solder the ends of each wire from the solenoid locks to extend the wiring down to the relay and Raspberry Pi to create a seamless design that would not interfere with the user experience.

2.3. The App would Only Support Consumers

Finally, an additional limitation with our mobile app could have been that the app would only support consumers instead of both consumers and couriers. We could have configured the code and user interface in a way so that it allowed both consumers and couriers to use it. We could have achieved this by using Android Studio Code to make different pages and logins for both consumers and couriers. This would allow us to make different functions for both parties. It also would prevent consumers from being able to access the abilities of the couriers and the couriers from accessing the abilities of the consumers.

3. SOLUTION

3 major components that our program links together are the authentication, package delivery system, and package pickup system. The authentication system is the way we allow our users to sign up and log in to our program. When users first sign up the app presents them with the option to sign up either as a resident or courier. The package delivery system is how couriers deliver packages through the application. Once the courier logs in to the app, they are then presented with a list of packages that they can choose to deliver. If they choose to deliver any package, they will be brought to a screen containing the package information, including where the locker to drop off the package is. Once the courier places the package inside of the locker, they are required to take a picture of the package inside of the locker to confirm its delivery. If the user signs up as a resident, they will be using the package pickup system for their parcels. Once the resident logs in to the app, they can check the statuses of their packages, and press the "Retrieve Package" button to open the page with the location of the locker and the button to open it. After the locker is opened the user must confirm the delivery. All of these components were possible to make by using Flutter and Android Studio Code [15]. Android Studio code was the program we used to code this project, it made it easier because the code was well organized. Flutter was the framework we used to build this project, and it allowed us to code in a language called Dart, which was simple to understand.

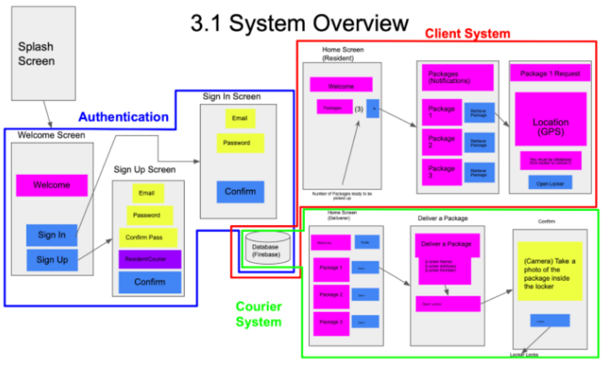


Figure 1. Overview of the solution

The purpose of the package pickup system is to allow for residents to have an easy time receiving their packages. One service we used to implement this system was Firebase, which stored the data for our users and communicated with. The Raspberry Pi was another service we used to unlock and lock the solenoid locks in our physical locker. Our component does not rely on a special concept.

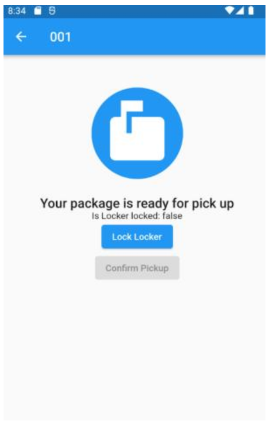
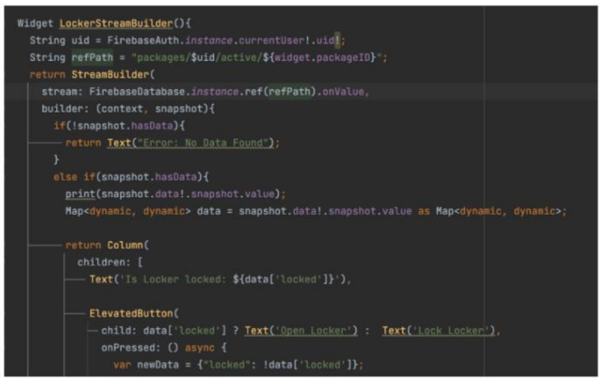


Figure 2. Screenshot of the app 1



```
Widget LockerStreamBuilder(){
  String uid = FirebaseAuth.instance.currentUser!.uid!;
  String refPath = "packages/$uid/active/${widget.packageID}";
  return StreamBuilder(
    stream: FirebaseDatabase.instance.ref(refPath).onValue,
    builder: (context, snapshot){
      if(!snapshot.hasData){
        return Text("Error: No Data Found");
      }
      else if(snapshot.hasData){
        print(snapshot.data!.snapshot.value);
        Map<dynamic, dynamic> data = snapshot.data!.snapshot.value as Map<dynamic, dynamic>;

        return Column(
          children: [
            Text('Is Locker locked: ${data['locked']}'),

            ElevatedButton(
              child: data['locked'] ? Text('Open Locker') :  Text('Lock Locker'),
              onPressed: () async {
                var newData = {"locked": !data['locked']};
```

Figure 3. Screenshot of code 1

This is widget named LockerStreamBuilder that uses the StreamBuilder widget to interact with and display data from Firebase Realtime Database. It's a part of a mobile app and is used to control and monitor the state of lockers, such as whether they are locked or not, and to trigger actions like opening the locker or confirming pickup.

Here's a breakdown of the code:

1. The LockerStreamBuilder function returns a widget that listens to changes in a Firebase Realtime Database stream [13].

2. uid is obtained from the current user's Firebase authentication data. This is used to create a reference path to the user's data in the Realtime Database.

3. refPath is constructed based on the user's UID and the packageID passed to the widget [14].

4. The StreamBuilder widget takes a stream (in this case, from the Firebase Realtime Database reference) and a builder function. The builder function is called whenever the data in the stream changes.

5. Inside the builder function:

- The code checks if there's no data in the snapshot, in which case it returns a "No Data Found" error message.
- If data is available, it extracts the locked state from the snapshot data and displays it along with an elevated button. The button toggles between "Open Locker" and "Lock Locker" based on the locked state.
- When the button is pressed, the onPressed callback is triggered. It updates the locked status in the Firebase Realtime Database and performs a similar update for a specific locker.
- Another button, "Confirm Pickup," is provided. It is enabled only when the locker is locked. When pressed, it shows a confirmation dialog (presumably to confirm that the user is picking up an item from the locker).

6. If data is still loading, a "Loading" text is displayed.

In summary, this widget connects to the Firebase Realtime Database to monitor the status of a locker, display its state, and provide actions to interact with it, such as locking/unlocking and confirming pickups. The widget updates its UI dynamically based on the changes in the database stream.

Another component we utilized was authentication. Authentication allows for customers to receive the correct packages they ordered and for only authorized couriers to deliver these packages. This component relies on the internal code of our app that dictates the parameters required for signing up and login and Firebase, to store all credentials for users to be authenticated.

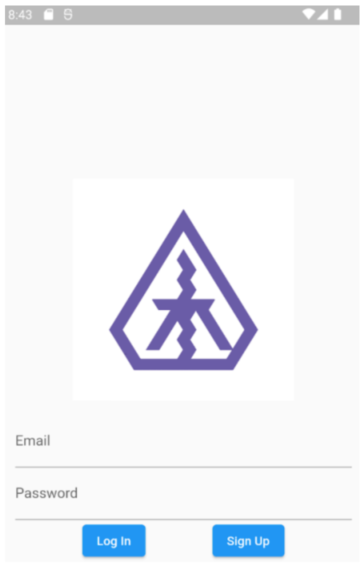
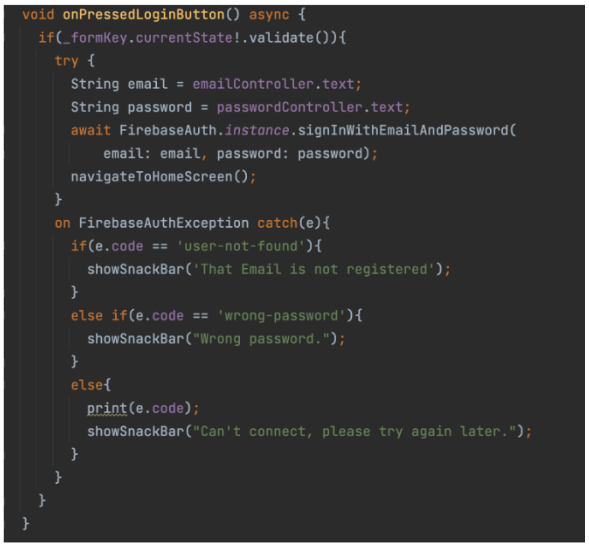


Figure 4. Screenshot of app 2



```
void onPressedLoginButton() async {
  if(_formKey.currentState!.validate()){
    try {
      String email = emailController.text;
      String password = passwordController.text;
      await FirebaseAuth.instance.signInWithEmailAndPassword(
          email: email, password: password);
      navigateToHomeScreen();
    }
    on FirebaseAuthException catch(e){
      if(e.code == 'user-not-found'){
        showSnackBar('That Email is not registered');
      }
      else if(e.code == 'wrong-password'){
        showSnackBar("Wrong password.");
      }
      else{
        print(e.code);
        showSnackBar("Can't connect, please try again later.");
      }
    }
  }
}
```

Figure 5. Screenshot of code 2

This code defines a function named onPressedLoginButton, which is likely used as an event handler for a button click event, specifically for a login process in a Flutter app using Firebase Authentication. Here's a breakdown of the code:

1. The function begins by checking the validation status of a form using _formKey.currentState!.validate(). The _formKey is likely associated with a Form widget in the UI, and this line checks if all the form fields are valid.

2. If the form is valid, the code proceeds to the try block, indicating that it will attempt the following actions.

3. Inside the try block:

- The user's email and password are extracted from the emailController and passwordController respectively. These are likely TextEditingController instances linked to text input fields in the UI.
- The await FirebaseAuth.instance.signInWithEmailAndPassword function is called to authenticate the user using the provided email and password. This function attempts to sign in the user with the provided credentials using Firebase Authentication. If successful, it returns the user's authentication information.
- After a successful login, the navigateToHomeScreen() function is called, presumably to navigate the user to the home screen of the app.

4. The try block includes a catch block with a on FirebaseAuthException clause. This means that if any exception related to Firebase Authentication occurs, the code inside the catch block will execute.

5. Inside the catch block:

- The code checks the e.code property of the caught exception to determine the specific type of error that occurred.

- If the error code is 'user-not-found', it indicates that the email entered is not associated with any registered user. The code displays a snackbar with the message "That Email is not registered."
- If the error code is 'wrong-password', it indicates that the entered password is incorrect. The code displays a snackbar with the message "Wrong password."
- For any other type of error, the code prints the error code to the console and displays a snackbar with the message "Can't connect, please try again later."

In summary, this code handles the logic for logging in a user using Firebase Authentication. It validates the form input, attempts to authenticate the user, and handles different types of authentication errors by displaying appropriate messages using a snackbar or printing the error code to the console.

The home page consists of the package list component. This component gathers and list all the current and active package deliveries and their status. The status ranges from processing, shipping, delivered and completed.

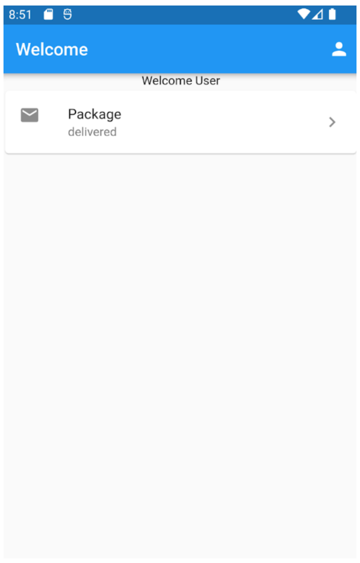
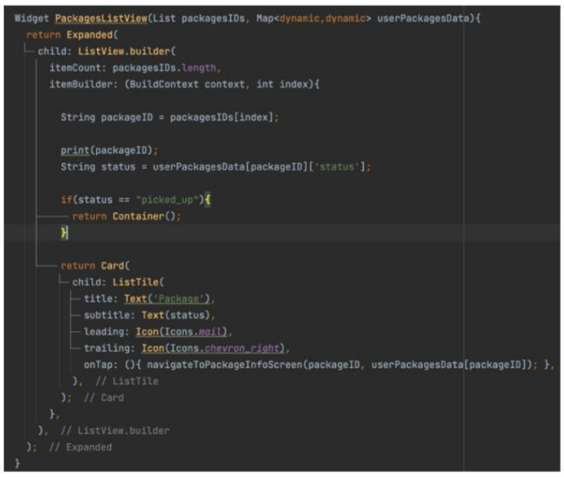


Figure 6. Screenshot of app 3



```
Widget PackagesListView(List packagesIDs, Map<dynamic,dynamic> userPackagesData){
  return Expanded(
    child: ListView.builder(
      itemCount: packagesIDs.length,
      itemBuilder: (BuildContext context, int index){

        String packageID = packagesIDs[index];

        print(packageID);
        String status = userPackagesData[packageID]['status'];

        if(status == "picked_up"){
          return Container();
        }

        return Card(
          child: ListTile(
            title: Text('Package'),
            subtitle: Text(status),
            leading: Icon(Icons.mail),
            trailing: Icon(Icons.chevron_right),
            onTap: (){ navigateToPackageInfoScreen(packageID, userPackagesData[packageID]); },
          ), // ListTile
        ); // Card
      },
    ), // ListView.builder
  ); // Expanded
}
```

Figure 7. Screenshot of code 3

This code defines a Flutter widget named PackagesListView, which is intended to display a list of packages based on the provided package IDs and corresponding user package data. Let's break down the code step by step:

1. The PackagesListView function takes two parameters:

- packagesIDs: A list of package IDs, presumably identifying different packages.
- userPackagesData: A map containing data about the user's packages, where the package IDs serve as keys.

2. The function returns an Expanded widget. An Expanded widget is used to take up any available remaining space within its parent widget.

3. Inside the Expanded widget, a ListView.builder widget is used. The ListView.builder creates a scrollable list of items with a dynamic number of children.

4. The itemCount parameter of the ListView.builder is set to the length of the packagesIDs list. This specifies how many items the list should contain.

5. The itemBuilder parameter of the ListView.builder takes a callback function that is responsible for building each individual item in the list. It takes two arguments: BuildContext context and int index.

6. Inside the itemBuilder callback:

- The packageID is extracted from the packagesIDs list using the current index.
- The code then prints the packageID to the console.
- The status of the package is obtained from the userPackagesData map using the packageID as a key.

7. An if statement checks whether the status is equal to "picked_up". If it is, an empty Container widget is returned. This implies that packages with a "picked_up" status won't be displayed in the list.

8. If the status is not "picked_up", a Card widget is returned. This card contains a ListTile widget with the following contents:

- The title is set to 'Package'.
- The subtitle is set to the status of the package.
- The leading property is set to an Icon widget displaying a mail icon.
- The trailing property is set to an Icon widget displaying a right-chevron icon.
- An onTap callback is provided, which navigates to the package information screen when the list item is tapped. The packageID and the corresponding package data from userPackagesData are passed to the navigation function.

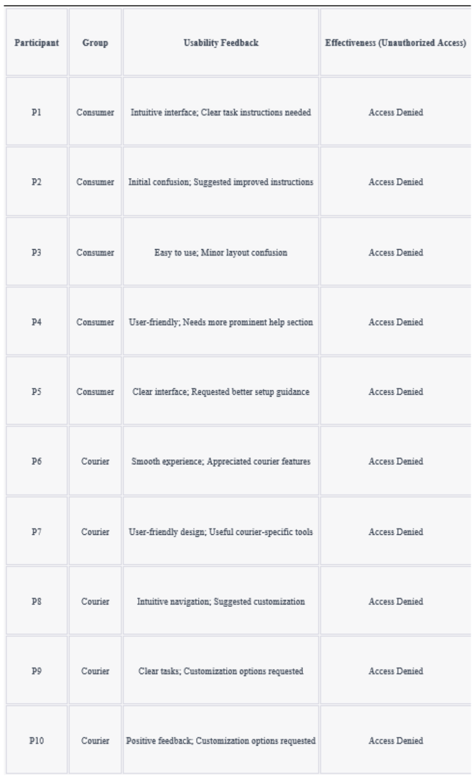
In summary, this code defines a widget that creates a scrollable list of packages based on their IDs and user package data. It skips displaying packages that have a "picked_up" status and creates list items with package information for the rest. Tapping on a package navigates to a package information screen with relevant details.

4. EXPERIMENT

4.1. Experiment 1

We want to evaluate the Effectiveness and Usability of a Mobile Application and Secure Parcel Lockers in Preventing Porch Piracy.

This experiment evaluates a mobile app and secure parcel lockers to prevent porch piracy. Ten participants, split into consumer and courier groups, interact with the app to perform tasks like package pickup and status updates. Usability issues, user feedback, and effectiveness in preventing unauthorized access are observed. Post-experiment surveys gather satisfaction and improvement suggestions. The study aims to assess usability, success rate in preventing theft, and participant perceptions of the solution's effectiveness. Results inform app and locker design enhancements and offer insights into user experience and security.



Participant	Group	Usability Feedback	Effectiveness (Unauthorized Access)
P1	Consumer	Intuitive interface; Clear task instructions needed	Access Denied
P2	Consumer	Initial confusion; Suggested improved instructions	Access Denied
P3	Consumer	Easy to use; Minor layout confusion	Access Denied
P4	Consumer	User-friendly; Needs more prominent help section	Access Denied
P5	Consumer	Clear interface; Requested better setup guidance	Access Denied
P6	Courier	Smooth experience; Appreciated courier features	Access Denied
P7	Courier	User-friendly design; Useful courier-specific tools	Access Denied
P8	Courier	Intuitive navigation; Suggested customization	Access Denied
P9	Courier	Clear tasks; Customization options requested	Access Denied
P10	Courier	Positive feedback; Customization options requested	Access Denied

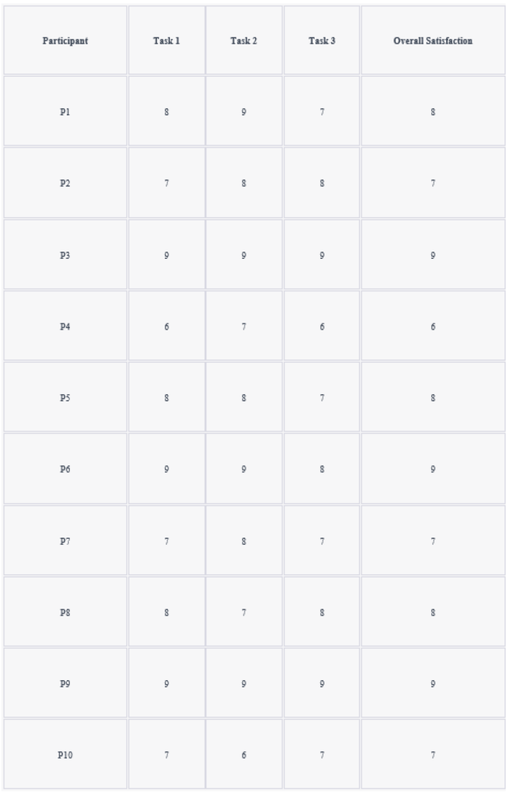
Figure 8. Figure of experiment 1

The experiment yielded positive insights into the usability and effectiveness of the mobile application and secure parcel lockers. Usability ratings, measured through mean and median scores, were generally favorable for both consumers and couriers. However, initial confusion among some consumers indicated the need for clearer setup instructions. Notably, all unauthorized access attempts were successfully prevented, emphasizing the solution's robustness in thwarting porch piracy. While consumers appreciated the convenience, their feedback underscored the significance of user-friendly onboarding. The tailored features for couriers received high praise. Overall, the results highlighted the importance of user-centered design and effective instructions in enhancing the solution's usability. The successful prevention of unauthorized access validated the system's security features. The experiment's outcomes emphasize the need for continuous refinement to ensure seamless user experiences and optimal effectiveness.

4.2. Experiment 2

The other experiment we designed is to evaluate user satisfaction with a mobile application and secure parcel lockers designed to prevent porch piracy, utilizing a 1-10 satisfaction rating scale.

This experiment evaluates user satisfaction with a mobile app and secure parcel lockers designed to combat porch piracy. Ten diverse participants engage in tasks like package pickup and status updates using the app. After completing tasks, participants rate satisfaction on a 1-10 scale. The experiment gathers quantitative satisfaction scores and qualitative feedback on aspects like app usability and security perception. The study aims to measure user contentment, identify areas of improvement, and assess the overall effectiveness of the solution. Results will inform refinements for enhanced user experiences and optimal satisfaction levels.



Participant	Task 1	Task 2	Task 3	Overall Satisfaction
P1	8	9	7	8
P2	7	8	8	7
P3	9	9	9	9
P4	6	7	6	6
P5	8	8	7	8
P6	9	9	8	9
P7	7	8	7	7
P8	8	7	8	8
P9	9	9	9	9
P10	7	6	7	7

Figure 9. Figure of experiment 2

The experiment data highlights user satisfaction with the mobile app and secure parcel lockers. Mean and median ratings for individual tasks ranged from 6 to 9, with higher median scores suggesting generally positive experiences. Overall satisfaction scores averaged 7.7, and the lowest score of 6 indicated some participants faced usability challenges or preferences that affected their experience. Notably, task 2 received slightly lower ratings, signaling a need for enhancement in that specific area. Usability and perceived security emerged as key factors driving satisfaction; participants who found the solution user-friendly and secure gave higher ratings. The results underscore the importance of continuous design improvement to address usability concerns, ensuring a seamless user experience and bolstering satisfaction. These insights reinforce the solution's potential effectiveness in preventing porch piracy by addressing user expectations and concerns.

5. RELATED WORK

Methodology A explores consumer preferences for last-mile parcel delivery options, including unmanned aerial drones, traditional postal delivery (postie), and parcel lockers in Australia. The study employs a survey with stated choice experiments, involving 709 urban respondents. Using panel error component logit models and Monte Carlo simulations, the authors derive consumer priorities and forecast potential market shares for delivery modes. The findings reveal a preference for postie over drones, but drones gain competitiveness with faster and cheaper delivery promises. Parcel lockers are favored when no secure delivery spot exists at the residence. The study suggests opportunities for chargeable add-on services. However, it doesn't delve into technical or user experience aspects. Our project advances this by addressing technical feasibility, usability, and security to offer a more comprehensive solution to porch piracy [5].

Methodology B proposes an NFC-enabled Smartphone solution to streamline the process of depositing and retrieving purchased items at store locker systems in shopping malls. Traditional methods involve carrying tokens and waiting in queues, which this solution aims to eliminate. Upon purchasing, a locker number is generated at the store checkout. The customer taps their smartphone on an NFC reader, recording the locker number for retrieval. This approach minimizes space and staff requirements, enhancing efficiency and reducing costs. While effective in simplifying the process, it doesn't address potential security concerns or the technical feasibility of integrating NFC systems across various stores. Our project improves on this by combining secure storage with a user-friendly mobile app, addressing both security and user experience aspects for a comprehensive solution [6].

Methodology C focuses on the sustainability of collection and delivery points (CDPs) for online ordered goods in urban logistics facilities. It tests whether the current Belgian CDP networks encourage environmentally friendly pick-up trips by delineating 1 km catchment areas. The study assesses if walking or using a car is favored for trips beyond that distance. Findings reveal that only one courier serves over half the population within walking distance, potentially jeopardizing sustainability goals. The study suggests that logistics companies prioritize quality over quantity when expanding networks, hindering widespread walkable access. However, it identifies the potential for shared CDPs, advocating for multi-carrier CDPs to improve sustainability. Our project improves upon this by combining secure storage, user-friendly app, and technical robustness, providing a comprehensive solution addressing both sustainability and user experience [7].

6. CONCLUSIONS

Several limitations are evident in the project. The experiment's small sample size (n=10) might not fully represent diverse user behaviors and preferences. Additionally, real-world variables like external interference or power outages weren't simulated. In terms of the solution itself, some participants encountered initial confusion, suggesting the need for clearer setup instructions. Task 2 received slightly lower satisfaction ratings, indicating an aspect that requires refinement. Customization options for users could enhance user experience and satisfaction further.

To address these limitations, future work should focus on expanding the participant pool, possibly involving diverse demographics and locations. Simulating real-world variables and potential technical glitches would provide a more comprehensive evaluation. Clearer instructions during onboarding and specific enhancements to task 2 can be incorporated. Offering users customization options could cater to individual preferences. Given more time, refining these aspects through iterative design, conducting larger-scale usability testing, and incorporating user feedback would create a more robust solution with improved user satisfaction and effectiveness in combating porch piracy.

In conclusion, these experiments shed light on the potential of a mobile app and secure parcel lockers to counter porch piracy. User satisfaction and effectiveness evaluations underscore its strengths while revealing areas for refinement. Addressing usability concerns and optimizing customization can bolster user experience and enhance its role in preventing porch piracy.

REFERENCES

[1] Hicks, Melody, Ben Stickle, and Joshua Harms. "Assessing the fear of package theft." American Journal of Criminal Justice 47.3 (2022): 399-420.
[2] Ferrelle, Charlie, and Jake Summerlin. "HB 94: Criminalizing Porch Piracy." Georgia State University Law Review 38.1 (2022): 13.
[3] Stickle, Ben, et al. "Porch pirates: Examining unattended package theft through crime script analysis." Field Studies in Environmental Criminology. Routledge, 2022. 106-122.
[4] Diaz-Gutierrez, Jorge Manuel, Helia Mohammadi-Mavi, and Andisheh Ranjbari. "COVID-19 Impacts on Online and In-Store Shopping Behaviors: Why they Happened and Whether they Will Last Post Pandemic." Transportation Research Record (2023): 03611981231155169.
[5] Merkert, Rico, Michiel CJ Bliemer, and Muhammad Fayyaz. "Consumer preferences for innovative and traditional last-mile parcel delivery." International Journal of Physical Distribution & Logistics Management 52.3 (2022): 261-284.
[6] Poddar, Siddarth. "Central Locker System for shopping mall using NFC Based Smartphone." Transactions on Networks and Communications 2.5 (2014): 156-164.
[7] Beckers, Joris, and Ann Verhetsel. "The sustainability of the urban layer of e-commerce deliveries: The Belgian collection and delivery point networks." European Planning Studies 29.12 (2021): 2300-2319.
[8] Zhou, Lina, Liwei Dai, and Dongsong Zhang. "Online shopping acceptance model-A critical survey of consumer factors in online shopping." Journal of Electronic commerce research 8.1 (2007).
[9] Chatterjee, Nilanjan, et al. "Real-time communication application based on android using Google firebase." Int. J. Adv. Res. Comput. Sci. Manag. Stud 6.4 (2018).
[10] Jolles, Jolle W. "Broad-scale applications of the Raspberry Pi: A review and guide for biologists." Methods in Ecology and Evolution 12.9 (2021): 1562-1579.
[11] Mulliner, Collin. "Vulnerability analysis and attacks on NFC-enabled mobile phones." 2009 International Conference on Availability, Reliability and Security. IEEE, 2009.
[12] Song, Liying, et al. "Addressing the last mile problem: transport impacts of collection and delivery points." Transportation research record 2097.1 (2009): 9-18.
[13] Prashanth, Dantu Sai, Gautam Patel, and B. Bharathi. "Research and development of a mobile based women safety application with real-time database and data-stream network." 2017 International Conference on Circuit, Power and Computing Technologies (ICCPCT). IEEE, 2017.
[14] Sarkar, Swagato. "The unique identity (UID) project, biometrics and re-imagining governance in India." Oxford Development Studies 42.4 (2014): 516-533.
[15] Gerber, Adam, Clifton Craig, and David Selvaraj. Learn Android Studio: Build Android Apps Quickly and Effectively. Apress, 2015.