Jaeseok Choi, Brian Morgan, Ahmet Yilmaz

Final Report

- **Section 1**: Team member names, and a description (a list) of who-did-what in the project.

  Ahmet Yilmaz:

  1. User Interface Functionalities:
     a. Sign up Activity to create an account
     b. Login Activity
     c. Rent Functionality:
        i. Add rental to the database
        ii. Find matching loaners
        iii. Show lockers having those loaners on Google maps
        iv. Associate the chosen locker with the rental
        v. Generate QR code to access the locker
     d. Loan Functionality:
        i. Add locker to the database
        ii. Find matching lockers
        iii. Show lockers having those loaners on Google maps
        iv. Associate the chosen locker with the loaner
        v. Generate QR code to access the locker
     e. Return Functionality:
        i. Update the database to make:
           1. The loaned item available again
           2. The locker available again
  2. User Interface has been Implemented using:
     a. Android Studio, Kotlin
     b. Firebase database for authentication
     c. Firebase real time database to store locker, rental, loaner objects
     d. Google maps to show locker locations or the matching loaners

  Jaeseok Choi:

  1. Lock Mechanism
     a. QR code recognition
     b. Locking & unlocking of solenoid
     c. Communication between the server and the Raspberry Pi
     d. Communication between the Raspberry Pi and another Raspberry Pi designated for object recognition

  Brian Morgan:

  1. Example AnyShare locker

    a. Wood construction
2. Object detection
    a. Using Tensorflow
    b. Differentiate detected objects
    c. Utilize Arduino to illuminate locker
    d. Communicate between other Raspberry Pi to validate item inside

- **Section 2**: A brief non-technical overview of your project (2 paragraphs).

Anyshare app will be used by anyone who would like to share a tool/gadget at our platform either to loan or to rent. Loaners are the users who would like to make money by renting the tools/gadgets they own. Renters are the users who would like to save money by renting tools/gadgets they need rather than purchasing. As minimalism is becoming more important in our lives, due to the high cost of living and high rate of relocation, anyone who does not want to make a permanent commitment to a tool/gadget that will be used only a few times a year will find our app appealing. Some users will want to be both renters and loaners at the same time to make money with the tools/gadgets that they already have but they do not use often, and to save money on the tools/gadgets that they need. Our app will also be attractive to the renters who would like to try out a product that they have never used before. As our app promotes utilization efficiency of the existing products and dampens consumerism, people who are concerned about environmental issues will want to use our app too. The app will also fill a gap as an alternative to online shopping. When a person is in urgent need of a tool/gadget which is not available in a local store, ordering online will not be a viable option due to the time limitation. In some other cases, if the tool is too heavy and the shipping is not covered by the seller, the cost of the online purchase will increase dramatically. In such cases, Anyshare will look attractive to anyone as it is fast, convenient, and cheap.

- **Section 3**: The link to your project website, with a list of items on the website. See the description above on this page about what the project webpage needs to have.

  https://akutluhany.github.io/index.html

- **Section 4**: A complete list of libraries, packages and APIs that you used for the project. This is not inclusive of your own code.

User Interface:

1. Android Studio and Emulator for development
2. Kotlin
3. Google Maps API
4. Anko Library for Android threading
5. OkHttp Library
6. JSON
7. Firebase Realtime Database
8. Firebase Authentication

Locking Mechanism:

1. OpenCV (cv2)
2. Raspberry Pi GPIO
3. pyzbar (Python zbar library)
4. imutils VideoStream
5. Bluetooth
6. Firebase Realtime Database

Object Recognition:

1. TensorFlow
2. OpenCV
3. Raspberry Pi
4. Arduino
5. NeoPixel LED
6. Bluetooth

- **Section 5**: A brief *technical* overview of your project (2 paragraphs).

The User Interface has three major components: User account creation and signup, creating rent/loan requests, matching algorithms. User account creation is handled by generic user interface and credentials are kept track via Firebase Authentication. When a user creates a rent request, a Rent object is created and added to the Firebase Realtime Database. Then the matching algorithm searches the database for a loaned item available. The results, i.e. the location of lockers where these loaners are stored, are shown on the Google maps for the user to choose the most convenient location. Then the full address of the chosen locker and the QR code is provided to the renter so that the renter can go and access the locker to pick the item. When a loaner follows the loan process to make an item available for loan, a Loan object is created in the Firebase Realtime Database. The available lockers are shown on the Google maps for the user to choose the most convenient location. Then the full address of the chosen locker and the QR code is provided to the loaner so that the loaner can go and access the locker to place the item in the locker. A fixed number of lockers are kept in the Firebase Realtime Database, and initially all are marked "available". When a loaner is associated with a locker, the Loan object gets updated with the locker's unique name and the QR code and the Locker object gets updated to be "unavailable". When a Rent object gets associated with a Loan object, Loan object gets marked "unavailable". Marking lockers as available/unavailable as well as Loan objects help with the matching process.

The locking mechanism requires three main parts: evaluating QR codes, locking and unlocking of solenoid, and communicating with another Raspberry Pi. The first step is the QR code evaluation process. Users can use their QR codes to unlock the designated lockers. Each locker has its own 'qr' attribute on the database. By comparing the user's QR code with this locker's 'qr' attribute, the locker's lock will be unlocked if successful or will stay locked if not so. In both cases, the users will be notified of the lock status from the server. Once the evaluation process is completed, the Raspberry Pi will send a signal to

the other Raspberry Pi for object recognition. The object recognition process lasts about 10 seconds and will notify back the status of the object. More details about the object recognition process are described in the next paragraph. Once the object is verified, the lock will be locked again for the next service.

- **Section 6**: One "if I had to do this again" paragraph from each team member that contains *technical* lessons learned, and what would you have done differently if you know what you know now (but with exactly the same project). Examples could include using a different API, some other type of restructuring etc.

  Ahmet Yilmaz: If I had to do this again, I would get my team to sign on the UI design from the beginning before I start coding. Our HCL HW2 required us to design the UI after the 90% demo in March, and in my opinion it is too late to agree on the UI and make changes on it as UI design is enmeshed with the functionality and data exchanged between the actions. I would also want to experiment with SQLite instead of Firebase Realtime Database as it was very hard to locate an item and modify it with Firebase Realtime Database.

  Jaeseok Choi: If I had to do this again, I would not have wasted time spent on setting up Raspberry Pi with my laptop. I spent much time and effort on figuring out how to make a connection between the Pi and my laptop and carrying extra stuff like a monitor, a keyboard, a mouse, and many cords to connect those devices to the Raspberry Pi. This was due to the fact that I was not able to connect the Pi to my laptop using the school's secure WiFi. After purchasing an ethernet cable adapter, I can now simply connect the Pi to my laptop by making the Pi use the local server. If I figured this out earlier, I would have been able to invest my time on advancing the project even further. However, I learned a lot from my mistakes and errors and was able to develop more concrete knowledge in Raspberry Pi.

  Brian Morgan: If I had to do this again, I would have not underestimated the requirements for device communication. Working with PyBluez to communicate between the two Raspberry Pis seemed easy from the surface, but soon became somewhat of a hassle. The need for Bluetooth communication stemmed from the lack of multiple camera ports on the Raspberry Pi. In a second iteration, it would be helpful to fully understand the requirements of the project and the plan of action among the team rather than build a plan as the project progresses. In this case, a different type of single board computer, like the Raspberry Pi could be used that would negate the need for two Raspberry Pis.

- **Section 7**: Instructions for follow-on projects. Write a detailed set of instructions for the next group of SD students who choose to build on your project:
  - How to download and get the project working. If there's equipment, a description of where to purchase (what model # etc).
  - What works, what doesn't, what to be aware of (pitfalls, issues).
  - Ideas for next steps.

  Software:

1. User Interface
    a. Android Studio, packaged as an easily importable project directory
2. Intermediate Raspberry Pi
    a. Open CV
    b. Raspbian OS, latest version
    c.
3. Object Recognition Raspberry Pi
    a. Tensor Flow
    b. Arduino IDE
    c. Raspbian OS, latest version
    d. Windows for creating image set using LabelImg software
    e. Windows/Ubuntu for training model

Hardware:

1. User Interface
    a. Physical test device, not entirely necessary
2. Intermediate Raspberry Pi
    a. Raspberry Pi
    b. Pi Camera
    c. Solenoid
3. Object Recognition Raspberry Pi
    a. Raspberry Pi
    b. Pi Camera
    c. Arduino Uno
    d. NeoPixel LED light ring (camera illumination)
    e. Wood for construction of locker

Status:

1. User Interface
2. Intermediate Raspberry Pi
3. Object Recognition Raspberry Pi
    a. Object recognition works, accuracy can be improved
    b. Communication between Raspberry Pis and Arduino work

Ideas:

- Find a prefabricated box that would allow for easier expansion
- Find a way to avoid using an Arduino for the sole purpose of illuminating the camera inside (like maybe a webcam with a usb-controllable illumination area)
- Search for a Single Board Computer, like the Raspberry Pi that has the capability for multiple cameras to avoid 2 per locker
- Search for better options other than a solenoid

- Consider other database options since Firebase may become an expensive option if the project turns into a real business