EC 544 Project

Smart Doorbell

Github: <https://github.com/ziyu39076/Smart_DoorBell>

|  |  |
| --- | --- |
| Team Member | Email Address |
| Qianhao Liulin | linqianh@bu.edu |
| Ziyu Zhao | zepher@bu.edu |

1. Introduction

In the context of the Internet of everything, the conception “smart home” has penetrated into all aspects of people's life. For example, in the field of Intelligent Security closely related to consumers, intelligent security devices represented by smart cameras have become an important support in family security activities. As a very important part of family security, intelligent doorbell undoubtedly plays the role of "frontier sentry", adding a security guarantee to the family door.

We want to build a smart doorbell for modern family which is capable of automatically open the door for friends and family members and send alerts to the user if their home is visited by strangers. Our idea is that use an embedded system to capture and preprocess the photo of visitor, then send the photo to a web server which will classify whether this visitor is permitted to enter, if so, activate the door, if not, record this visit and send alert to the user, and the user could proceed with further operations such as call the police or simply keep the door closed.

Raspberry Pi with camera

AWS face recognition API

Web Server

Database

visitor

Send request

Send response

Call API

Utilize data in DB

Interact with DB

Figure 1: project flowchart

2. Resources

1. Hardware
2. Raspberry Pi 2 Model B
3. Raspberry Pi Camera Video Module
4. Software
5. Linux Ubuntu LTS 18.04
6. Python 3.6.9
7. Nginx web server
8. Gunicorn
9. Supervisor (a software in Linux enviornment)
10. Git
11. Flask Framework
12. Bootstrap
13. Javascript
14. CSS
15. HTML
16. Database (SQLite)
17. VSCode
18. PyCharm
19. Web Server
20. AWS

3. Organization and Structure

1. Hardware part
   1. Discussion of failure
   2. Technical report of success
2. Software part (Flask web app)
   1. Discussion of failure
      1. Fail to add reset password link for user in case a user forgets his/her password
      2. Fail to add delete permitted visitor functionality
      3. Fail to send alert email to user if a stranger visited the user’s home

(these unfinished functionalities could be finished with more time and efforts)

* 1. Technical report of success
     1. Developed a fully functional web app using Python with Flask framework for user to register, login, add permitted visitor and review visit records
     2. Successfully achieved interaction between web app and database using Flask-SQLAlchemy
     3. Implemented face recognition using AWS face-recognition API
     4. Managed project version and updated project iteratively with Git version control system
     5. Configured Linux server for hosting web app in AWS EC2 instance
     6. Deployed flask app on AWS EC2 instance with Nginx and Gunicorn
     7. Achieved web app auto-start and auto-restart with Supervisor

1. Project details
   1. Hardware
   2. Software
      1. Web app development
         1. In Flask app, each page is navigated by function decorated with routes, and each page is a rendered template with style defined in Bootstrap and local CSS file
         2. Date submission is handles with build-in Class called WTForm in Flask-WTF, forms could be passed into flask templates to be displayed in pages
         3. Date storage and manipulation is handled with built-in class called SQLAlchemy, each table is represented by a defined class and we do not need to write SQL commands when developing web app
         4. User session management is achieved with LoginManager imported from flask\_login
         5. In client side, the raspberry pi could simulate as a user with request lib in python then login and post images for web app to identify
      2. Server configuration
         1. Nginx web server is deployed to handle incoming requests
         2. Gunicorn is used to call python scripts to run our application
         3. Supervisor is utilized to achieving running our app in the background with auto-start and auto-restart feature
2. Learned skills
   1. Web app development with Python and Flask framework
   2. Database interaction and basic SQL commands
   3. Client server communication and basic URL and HTML parsing
   4. Basic HTML and CSS knowledge
   5. Calling external APIs and setting up credentials
   6. Version control with Git
   7. Linux web server configuration
3. References
   1. Flask tutorial videos:

<https://www.youtube.com/playlist?list=PLosiE80TeTs4UjLw5MM6OjgkjFeUxCYH>

* 1. Git official doc: <https://git-scm.com/book/en/v2>