TECHNICAL INSTALLATION MANUAL

Technical Requirements & Reasoning

Argus is a home security system capable of instantly identifying threats on multiple video feeds around the home of a user while notifying the user of identified threats. Every choice we made was based on whether or not the software/technology was free and Open Source!

Github

Link to Repo: https://github.com/COS301-SE-2020/Home-Security-System

IDE

Link:
https://www.jetbrains.com/idea/download/?gclid=EAIaIQobChMIuOatp6qo6wIV5IBQBh29oAufEAAYASABEgKO__D_BwE#section=windows

- Install IntelliJ Ultimate so that you have access to database functionality
- Create a new Maven project with python plugins

Java

Windows Link: https://www.oracle.com/za/java/technologies/javase-downloads.html

- Install Java SE 14

Linux command: sudo apt install openjdk-14-jdk

Heroku

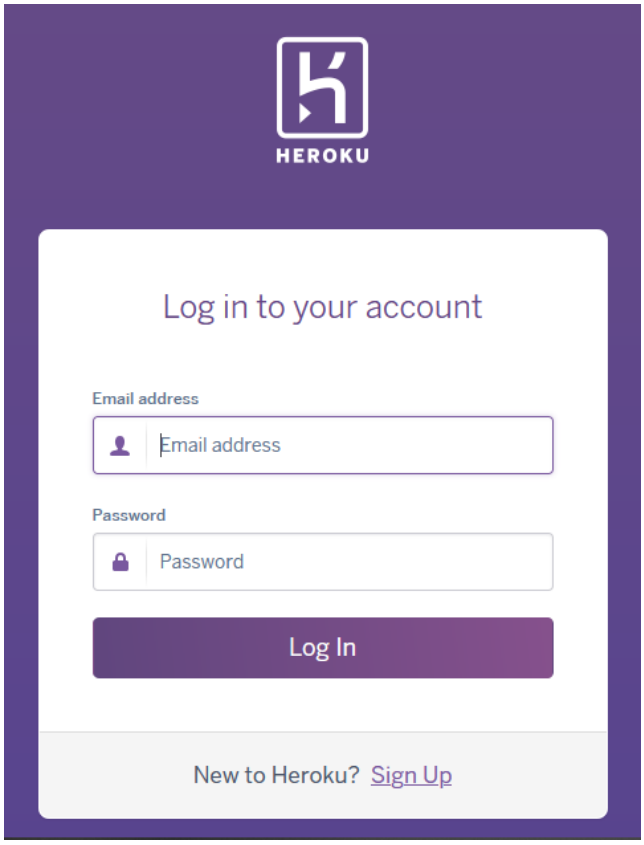
Link: https://id.heroku.com/login

```
npm install -g heroku
```

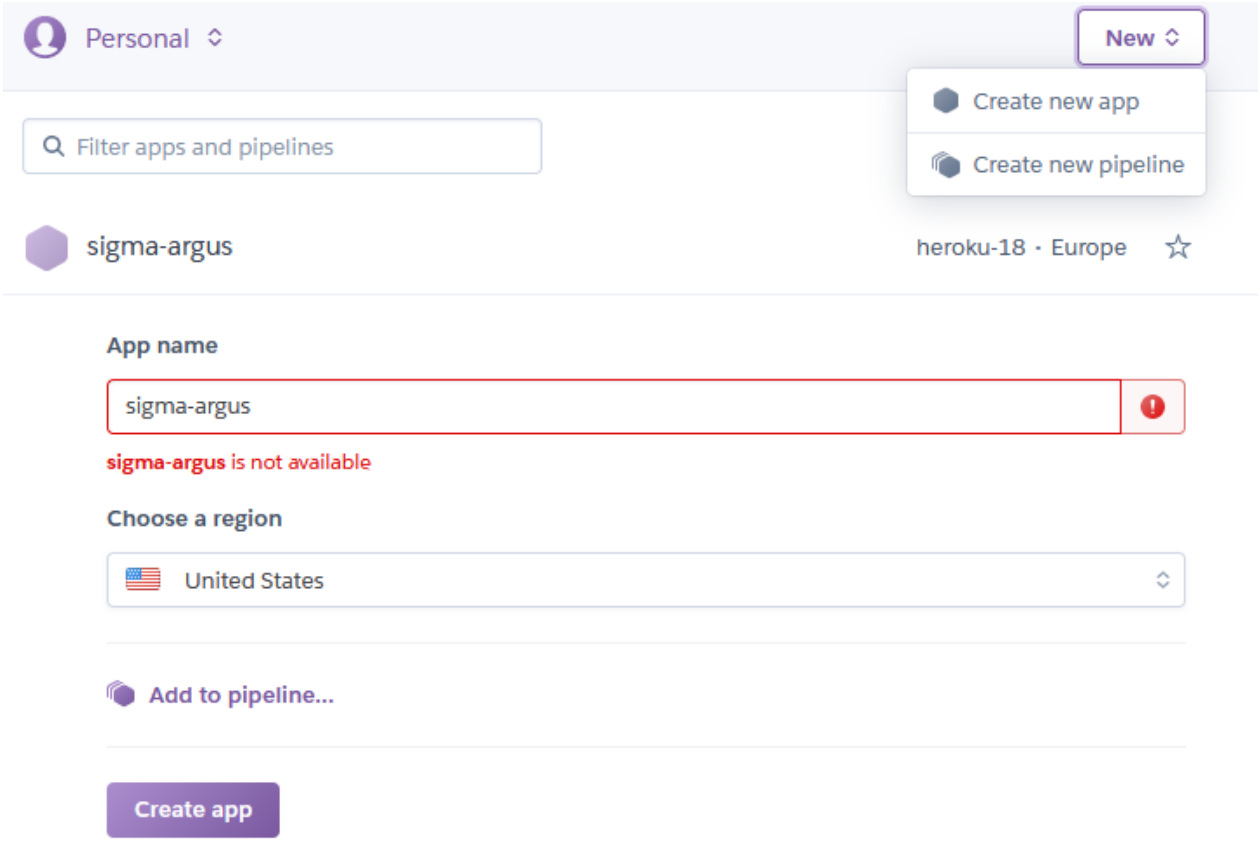
Reason: For our server we chose Heroku which is a free container-based cloud Platform as a Service for the deployment and management of apps. It is also largely scalable and provides free hosting services such as PostgreSQL & CloudAMQP (RabbitMQ).

Installation:

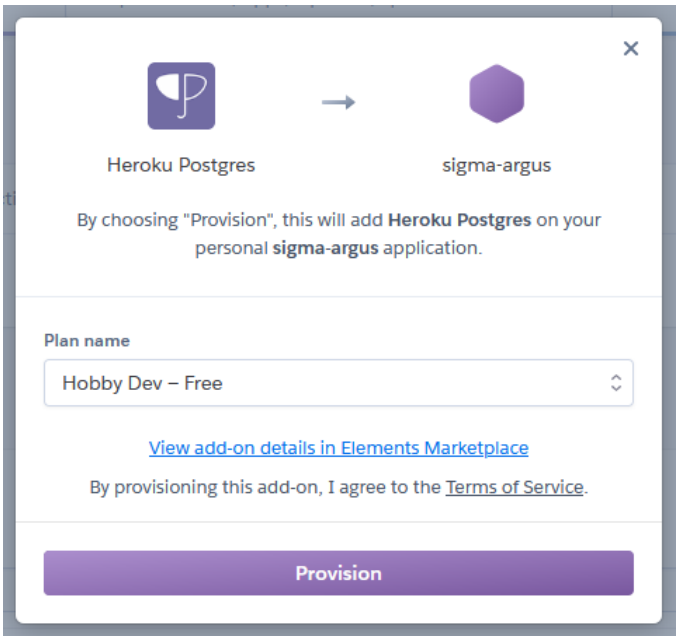
1. Create an Heroku account & Login



2. Create a new App



3. Create addons (Heroku PostgreSQL)



Angular/Node.js

Link: https://angular.io/

Link: https://nodejs.org/en/download/

- Latest version on any platform

Reason: For our Frontend we used Angular as it helps build interactive and dynamic single page applications (SPAs) with its compelling features including templating, two-way binding, modularization, RESTful API handling, dependency injection, and AJAX handling.

Installation:

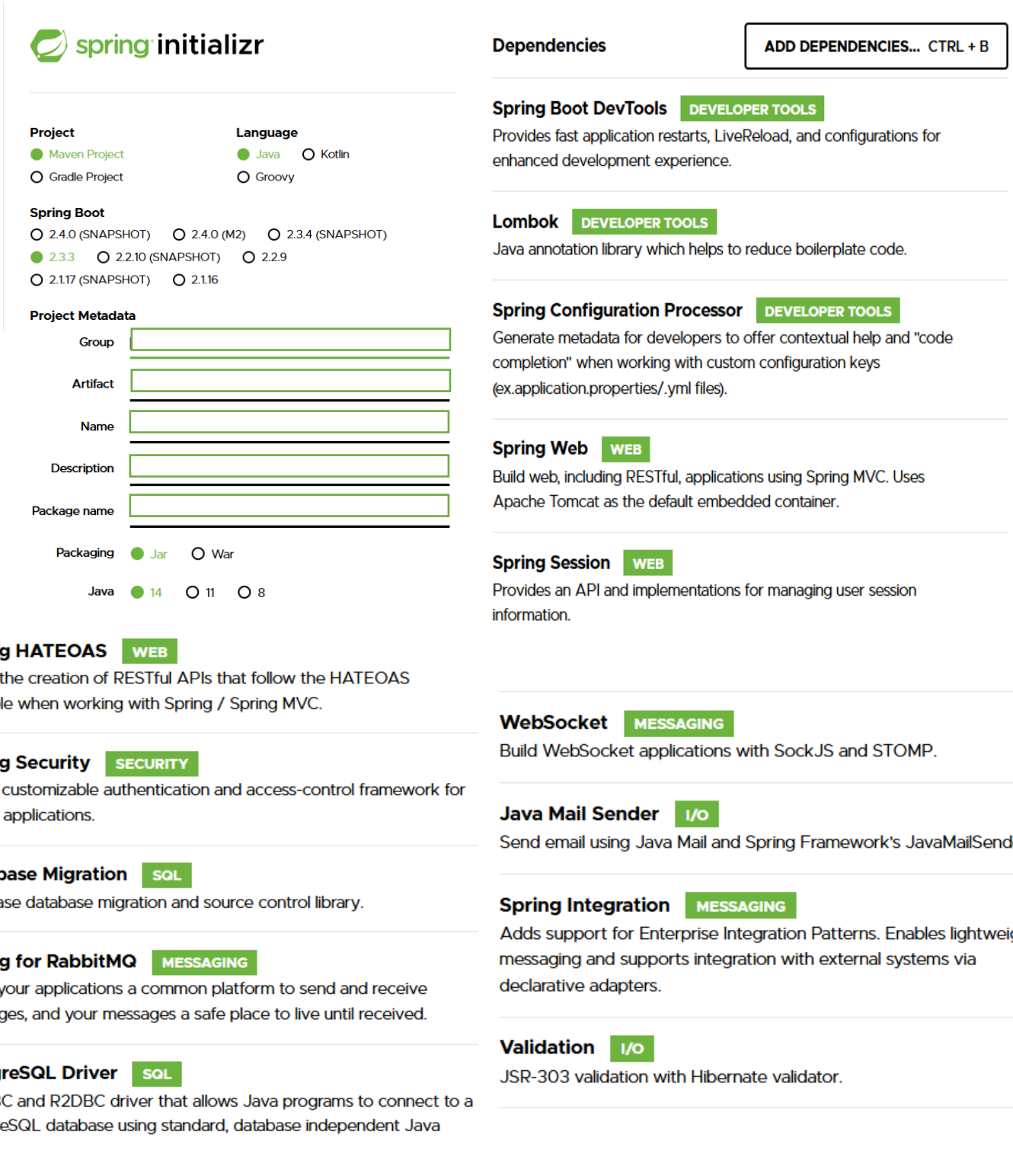
- npm *install* -g *@angular/cli*
- ng new [app]
- ng g c [component]
- ng g s [service]

Spring Boot

Link: https://start.spring.io/

Reason: Java Spring Boot framework was chosen as the Backend for our application as it is used to reduce overall development time and increase efficiency by having a default setup for unit and integration tests as well as provides built in libraries to handle CRUD operations and REST API's with the use of Hibernate and Controllers.

Installation: We used Spring initializer to create your Spring Boot project along with all the listed dependencies you see which will be added to your pom.xml file and configured automatically.



PostgreSQL (for local deployment)

Link: https://www.postgresql.org/download/

- Version 12
- Used for local storage (Not for production)

Reason: PostgreSQL was chosen as our database as MySQL or SQLite weren't as scalable nor widely used in production, that was another reason we chose to use is over something like Firebase as Firebase is normally only used for prototyping.

Installation: once installed open up pgAdmin4 and create a new User with the name "Sigma" password "Argus" and then create a new database called "argus_db" with "Sigma" as the owner. (Be advised this is if you wished to run the code without the server)

Liquibase

Link: https://www.liquibase.org/download (optional)

Reason: Liquibase was used to connect Spring Boot to either the local PostgreSQL database or the Heroku database Liquibase was chosen over Flyway as Flyway supports migration scripts in SQL and Java format only while Liquibase abstracts away from SQL completely and decouples database refactoring from the underlying database technology meaning that it is possible to change the database being used without having to change how the backend functions.

Installation: No installation required if dependency added to Spring Boot

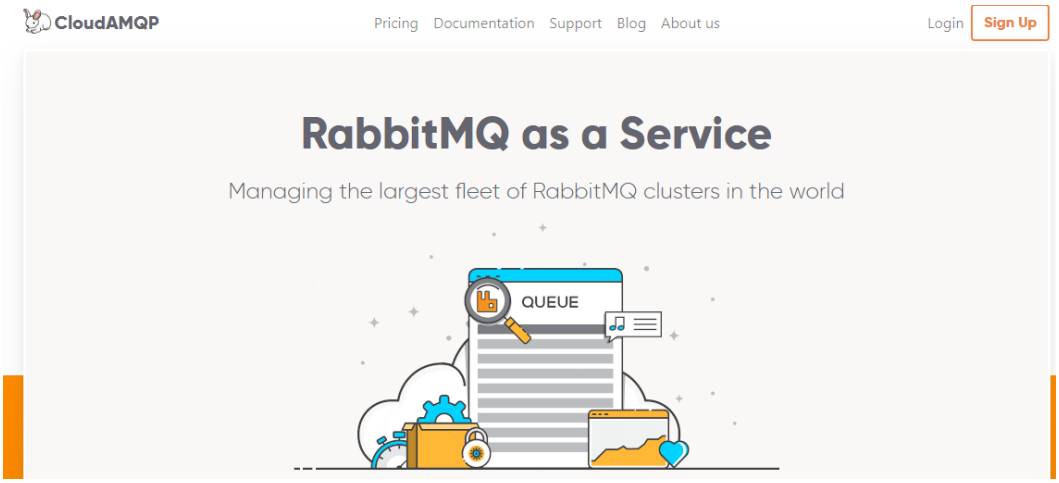
RabbitMQ Server & ERLANG (for local deployment)

Link: https://www.rabbitmq.com/download.html

Link: https://www.erlang.org/

Reason: RabbitMQ broker is user-friendly, easy to use especially with the Spring Boot framework. It's also scalable and flexible so as long as you don't exceed payload of the message object.

CloudAMQP (RabbitMQ Server)



Link: https://www.cloudamqp.com/

Reason: CloudAMQP is a free messaging server hosted online which provides everything RabbitMQ related, it can be connected through Heroku or via the URL provided to you when an instance is created. We chose not to use the add on as we were already using the PostgreSQL addon for the virtual database and due to Heroku charging for more than one add-on we chose not to use it.

Installation: No installation required just a dependency added to Spring Boot, Spring boot will connect automatically and manage your queues for you.

Python

Link: https://www.python.org/downloads/release/python-380/

Reason: There was a choice between Java (Nd4j) or Python (Tensorflow) for the backend Neural Network to do Facial Recognition, however; in the end we chose to go with Python for the Neural network as there are more libraries and resources to assist you with facial recognition as well as Python is lightweight and the Neural Network would need to run on a Raspberry Pi meaning it would only needed to communicate with the Spring Boot backend via a Message Queue.

Installation:

1. Download Python 3.8
2. Add IntelliJ's Python Addon
3. Finally install the following either through IntelliJ or via terminal

```
import cv2 as c
from mtcnn.mtcnn import MTCNN
import numpy as np
import tensorflow as tf
from scipy.spatial.distance import cosine
from keras_vggface.vggface import VGGFace
from keras_vggface.utils import preprocess_input
import os
import pika as pi
import time
import json
import base64 as b64
```

- OpenCV or cv2 for video footage
- MTCNN for the model
- Numpy storing files and feature comparison
- Keras_VGGFACE for facial classification
- Tensorflow used for the underlying model which the MTCNN model runs on
- Pika is used to create a connect to CloudAMQP(RabbitMQ)

The python code is running on a Raspberry Pi with multiple camera's attached. Your Application should just connect to the camera feed(s) automatically.

User Manual

Link: https://github.com/COS301-SE-2020/Home-Security-System/blob/master/Documentation/Argus_User_Manual.pdf