Title: Speech Radar

Name: Shah Ali 33455846

Abstract

Speech Radar is an Android application that allows users to locate their phone through speech recognition. The application uses Java, XML and Firebase Database (as backend) to achieve this. Python with TensorFlow module is used to train a model for speech recognition, which will be used to make predictions in the app.

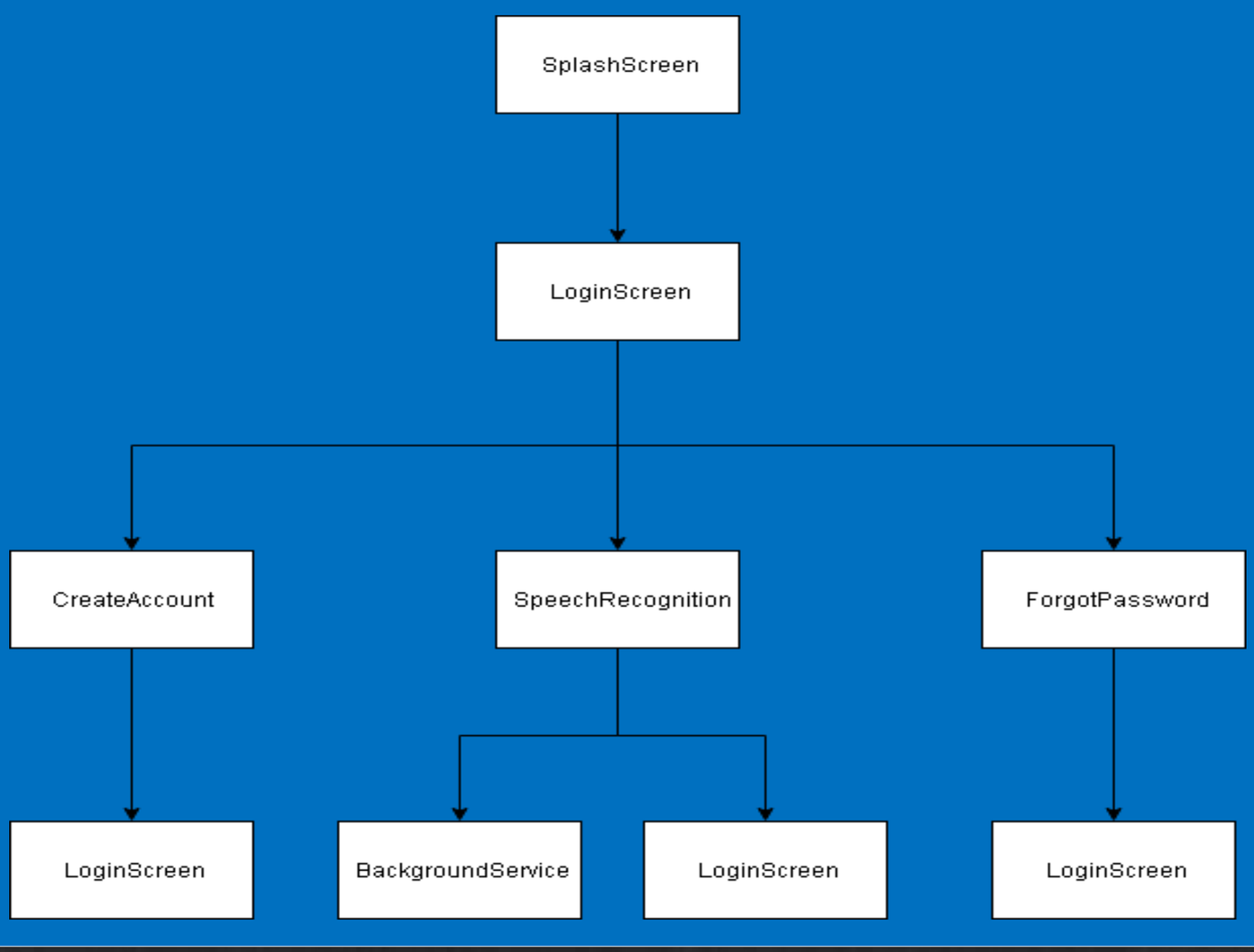
Introduction & Background

Suppose for instance an individual must rush to work and is unable to find their phone, but knows it's located somewhere inside their room. Speech Radar can speed up the search time just by an utterance of a specific word. Once the phone detects the utterance, it will begin ringing at maximum sound level, which will help the individual locate their phone.

Once app is activated, the continuous speech recognition service will run forever in the background and listen in for the code word. This service can be classified as state-of-the-art. The best example would be Google who use continuous speech recognition to listen in for the keyword 'Ok Google.'

The literature background goes over different articles relevant to this project. For example, one article describes PocketSphinx, a library that can be used in Java to ease implementation for the background service.

Hierarchy Diagram



Specification & Implementation

Main system requirements:

- Email verification sent when user creates account to prove that email address exists
- When activate button is pressed, background service should start
- Background service should deactivate when user utters the code word or terminates the app
- background service should not deeply affect performance of the phone (e.g. drain too much battery life)
- A backup of all user accounts should be kept in separate database in case of a disaster
- The app should be compatible on all Android devices running on API level 16 (Jelly Bean) or more

Implementation:

Firebase was used to store all accounts. Additionally, It helped in authenticating users in login screen, send reset password emails to users upon request and email verifications.

Unfortunately, implementation for TensorFlow didn't go according to plan. Models created had good testing accuracy and loaded into Android Studio. But, the audio class in Java did not work well with the model and thus, wrong detections kept being made. Thankfully, TensorFlow themselves created a model that worked well with this class. Since TensorFlow declared it as open-source, the same model was used for the project. References are made in the report.

Setting up the background service was eased by Android development tools. Adding the continuous speech recognition was tricky because PocketSphinx library was initially used for detecting speech. But the process of getting good detection rate was complicated and would require a long period of time to get right. Thus, Android's SpeechRecognizer class was used instead.

Testing & Evaluation

Software testing: described the quality of the app through black box and white box testing techniques.

User testing : gained feedback on the interface and speech recognizer by testing with 216 actual users

Self-experimentation: Personal testing helped investgate the maximum distance the background speech recognizer could detect code word (i.e. 6 metres)

Evaluation: Creating a recovery database was the only requirement not met. Thus, a recovery was made using SQLite. Also, issues were found in interface from user testing. For example, users expected login screen to remember their details instead of having to input their details every time they launched the app. This was immediately fixed.

Conclusions & Future Work

Project success: main success would be that all system requirements were met, suggesting how the project accomplished everything that was originally set out.

Project failure: main failure would be that most parts of the application require Internet connection (especially the speech recognizer.)

Similar apps: Find My Phone Whistle and Voice to Find My Phone

Future work: Build an iOS version of the app to expand target audience
Use the PocketSphinx library to create an offline background speech recognizer, so device can be found in situations where Internet is not available
Add cooler code words to make it less boring
Enable users to choose what sound they want to play when code word is uttered
Allow users to find phone by clapping or whistling
Allow users to deactivate or reactivate the service by uttering a specific word