Project Report - Void Network

Diogo Pedro
fc56337@alunos.fc.ul.pt
Faculdade de Ciências da Universidade de Lisboa
Lisboa, Portugal

Manuel Cardoso
fc56274@alunos.fc.ul.pt
Faculdade de Ciências da Universidade de Lisboa
Lisboa, Portugal

Ricardo Costa
fc64371@alunos.fc.ul.pt
Faculdade de Ciências da Universidade de Lisboa
Lisboa, Portugal

KEYWORDS

Mobile Computing, Android, Stranger Things, Upside Down, Cross-Dimensional Communication, Portals, Binary-Coded Signals, Image Detection, GPS, Camera, Push Notifications, Firebase, Supabase

1 INTRODUCTION

Void Network is an app that enables immersive, cross-dimensional communication in the Stranger Things universe, with binary-coded signal messages through Upside Down portals.

Stranger Things is a Netflix TV show with a unique fictional universe set in the 80's with a distinct division between the real world and the Upside Down. The Upside Down is a mysterious alternate dimension existing in parallel to the human world. This application allows users in this fictional world to communicate across both dimensions, blending location-based features with immersive communication.

This report outlines the concept, implementation and evaluation of this project, with the technologies, sensors and other device features used.

2 RELATED WORK

Void Network app draws inspiration from various existent applications and technologies:

- **Pokémon GO**: This game revolutionized mobile gaming by combining location-based mechanics where players explore real-world locations to interact with virtual elements.
- **WhatsApp/Discord**: These popular messaging platforms set the standard for seamless real-time communication through text, voice and video. Our app explores communication in a different way, aligning with the thematic context of Stranger Things.
- **Walkie-Talkie Apps**: Immediate, tactile communication inspired the manual signal messaging.
- **Stranger Things**: In the TV show, characters also communicate like this, through light signals, which inspired this idea altogether.

3 CONCEPT

This application is to be used in the Stranger Things universe, for users to communicate through dimensions with Upside Down portals. In the context of our app, portals are the way to allow communication between someone who is in the Upside Down and everyone else. If the user is in the real world, this restriction does not exist, otherwise, if the user is in the Upside Down, it is required to be within the range of a portal (1km radius), in order to send and receive messages.

The state that says whether the user is in the Upside Down or in the real world is dictated by the luminosity sensor - if the user is in the dark for too long, it is considered to be in the Upside Down, which is displayed by the change in theme, from light theme to dark theme, which clearly indicates this state, affecting the possible actions the user can take.

Portals can be registered using the camera and an image detection API. We consider a tree to be a portal. If one is detected, it is allowed to register the portal at the user's current location, with its name being the name of the street the user is in. These maps are then shown in a map along with its range and current user position, indicating if its in range or not. The portals listing are also shown, which can be inspected or navigate the map current's position to the portal's position.

The communication is based on binary-coded signals that can be translated based on user defined languages. These signals can be sent manually, using taps or light signals, or automatically, using the language dictionary directly. The signals are represented by short signal durations represented by '-' and long signal durations represented by '_'. These are used as message encoding signals, later translated into messages if that sequence of signals is present in the chosen language dictionary. The signals are then received as vibration and flashlight signals, which transmit the raw signals based on duration, along with push notifications, which include the signal meaning if existent. These messages can also be observed in the recent messages tab, indicating if it was sent or received, how long ago they were sent, and also a possibility of replaying the signal if clicked.

4 IMPLEMENTATION

In this section we describe the features, libraries, sensors and technologies used in the implementation of the application, as well as the features completed.

4.1 Architecture

The application was designed using the Model-View-ViewModel (MVVM) architecture, which provides a clear separation of concerns. This architecture ensures that the user interface remains responsive while handling complex operations such as real-time data updates, signal processing and location-based interactions. The view models manage the application's state and communicate with the views, ensuring a structured and maintainable codebase.

4.2 Organization

The codebase is well structured, and organized as follows:

- **/domain**: Contains core business logic, including data models and entities that define the application's primary functionality.
- **/repository**: Holds shared data among various components of the application.
- **/services**: Includes the foreground service implementation.
- **/storage**: Handles local storage operations, for saving and retrieving settings stored in the device with shared preferences.
- **/ui**: Contains the components related to the user interface.
  - **/navigation**: Manages navigation flow and routes within the app.
  - **/screens**: Defines individual screens and views.
  - **/theme**: Stores theme definitions for both light and dark theme.
  - **/utils**: Provides helper functions and utilities for the rest of the application.

4.3 Technologies

A range of technologies was used to implement this application:

- **Android**: The target platform for the app, along with their rich APIs.
- **Kotlin**: The programming language used for developing the app due to its modern features and seamless integration with Android.
- **Jetpack Compose**: Framework used to build the app's user interface.
- **Firebase Auth**: With anonymous user authentication, for distinguishing different users.
- **Firebase Real-time Database**: For storage of languages, portals and messages, with real-time synchronization.
- **Supabase**: Alternative to Firebase for file storage (portal photos), since Firebase Storage was removed from its free tier.
- **ML Kit**: For image labelling and detecting trees, which serve as portals. The library's pre-trained models efficiently identify tree-related features with a confidence threshold of 60%.
- **Mapbox**: To display the map with the user location and the registered portals and fetch street names.
- **Shared Preferences**: For storing user-defined settings locally on the device.

4.4 Sensors

The app uses several device sensors:

- **GPS**: For getting the user's current location, to display it in the map and get the distance from the closest portal.
- **Luminosity Sensor**: To detect light levels to toggle the Upside Down state and for building light signals.
- **Camera**: To capture images for validating portal registration, which are then processed using ML Kit to identify a tree.

4.5 Other Features

Additional device features were also used to implement the wanted type of communication:

- **Vibration**: For delivering received signals as vibration patterns corresponding to the encoded message.
- **Flashlight**: To transmit signals through light patterns, similarly to vibration.
- **Push Notifications**: Implemented via a foreground service to alert users of incoming messages. By using a foreground service, it is also possible to receive messages even if the app is closed. This service can be stopped in the app, silencing incoming signals.

5 EVALUATION

5.1 Development Process

From the pitch idea to the final application, there were some initial ideas that were later discarded, such as the 80's music player, which didn't really fit into the application and didn't make much sense to be implemented.

However, the main ideas were significantly enhanced and more well thought, such as the way of communicating, the addition of the foreground service with push notifications, and the way of registering portals with the camera and computer vision.

5.2 Project Guidelines

Overall, we were able to follow most of the guidelines for this project. It was designed as a mobile app due to its GPS requirements, which necessitate user movement to specific locations (e.g., near portals/trees) and interaction with the environment (e.g., taking photos to register portals). The app also leverages various sensors and APIs relevant to its context. The Firebase real-time database was pivotal for persistent storage and dynamic user interactions, enabling features like push notifications and event-triggered vibration and flashlight signals. Overall, the app integrates standard concepts into a unique, engaging package with innovative functionalities, which did not previously exist.

However, the user interface (UI) could be a bit more rich and interesting, but most importantly the user experience (UX) could be better with some of our users presenting some difficulty interacting and understanding the different functionalities of the app.

5.3 Contributions

The project was divided among the three of us, each with a core feature of the application: Ricardo with the communication aspect of the app and Firebase, Manuel with location features and the Mapbox map, and Diogo with the portal registration through the image detection API of ML Kit and photo storage through Supabase.

The overall estimate percentage each student contributed to the project is:

- Ricardo: 40%
- Manuel: 30%
- Diogo: 30%

6 CONCLUSION

To conclude, Void Network successfully bridges the gap between the fictional universe of Stranger Things and real-world technology, providing an innovative platform for immersive, cross-dimensional communication. By leveraging modern mobile technologies, sensors and APIs, the app delivers a unique experience that blends narrative-driven design with practical functionality.

Despite some areas for improvement, the project demonstrates how creative concepts can be realized through thoughtful implementation.

The source code of the project can be found at: https://github.com/VoidNetworkApp/VoidNetwork