The Human Interface Project

David Hampton n9135766

Asam Almohamed

I agree to the following:

1. I declare that all the work submitted for this assignment is my own original work except for material that is explicitly referenced and for which I have permission, or which is freely available (and also referenced).
2. I agree that QUT may archive this assignment for an indefinite period of time, and use it in the future for educational purposes including, but not limited to: as an example of previous work; as the basis for assignments, lectures or tutorials; for comparison when scanning for plagiarism, etc.
3. I agree to indemnify QUT and hold it blameless if copyright infringements are found in this work and the copyright owner takes action against QUT that is not covered by the normal terms of Educational Use.

David Hampton DH

**Preface:** Emerging evidence shows that the COVID-19 pandemic has had dramatic mental health impacts, particularly around loneliness. Estimates of the prevalence and distribution of social isolation and loneliness vary, ranging between 1-in-6 to 1-in-4 people (Smith & Lim, 2020). Social isolation and loneliness are understood to be distinct conditions, yet each has been found to predict premature mortality, depression, cardiovascular disease, and cognitive decline (Smith & Lim, [2020](https://www.phrp.com.au/issues/june-2020-volume-30-issue-2/how-the-covid-19-pandemic-is-focusing-attention-on-loneliness-and-social-isolation/)). It has also been shown to be as bad as high blood pressure, lack of exercise, obesity, or smoking ([Yang](https://www-taylorfrancis-com.ezp01.library.qut.edu.au/books/9781315148410), 2019). But loneliness is more than an individual problem, it is also a social problem. It can spread through a social network to up to 3-degrees of separation ([Cacioppo](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2792572/), 2009). In Australia, a national coalition – Ending Loneliness Together – has been established to bring together researchers and service providers to facilitate evidence gathering and the mobilisation of knowledge into practice (Smith & Lim, [2020](https://www.phrp.com.au/issues/june-2020-volume-30-issue-2/how-the-covid-19-pandemic-is-focusing-attention-on-loneliness-and-social-isolation/)).

# Project Objectives

The goal of this project is to build an anonymous and gamified cross-platform chat service that leverages the popularity of existing Big Chat Services (BCS).

Anonymity is achieved by using a third-party account to route messages between connected users. This third-party account parrots messages between connected users. If a user is not connected, the third-party account delivers message templates to elicit structured responses from the user. This helps the user navigate the backend using the chat interface as the front-end.

To do this the Raspberry Pi must host a program that delivers the following objectives:

1. Register webhooks and authenticate server-side platform clients
2. Listen to POST requests from webhooks and registration.
3. Expose the local server IP as a URL endpoint on the internet with a HTTPS tunnel.
4. Store and update user states based on POST requests.
5. Respond with API based on user state and message text

# Technology Selection (5 pages)

This section reviews and discusses the selected technologies selected to achieve these objectives and justify their selection.

## Technologies used:

The technologies work together to create a web server that transforms POST requests from webhooks into POST requests to an API. The language used is JavaScript which compiles JIT into native machine code using Google’s V8 ([V8](https://v8.dev/blog/ignition-interpreter), 2016) via the NodeJS runtime environment ([this](https://learning.oreilly.com/library/view/web-development-with/9781785287527/ch01.html#ch01lvl2sec07), 2015). The Node Package Manager CLI is used to easily integrate open source solutions to the required core functionality.

Express is a package that abstracts away the complexity of writing HTTP handling. It offers easily integratable middleware that responds to specific scenarios. Middleware used is Restify, CORS, and bottender, morgan. Restify and CORS allow the system to expose data at a URL endpoint which will be picked up by an admin panel hosted at Netlify. Morgan allows for automatic logging of all requests. Bottender handles responses to webhook urls and directs to

Bottender is a package that abstracts away webhook registration and client authentication (same for some platforms).

the authentication and webhooks from each platform. It also provides API access to the **messaging-api node module** which

V8 provides the

**NodeJS** - Node.js is a runtime environment for JavaScript built on top of Google's V8 engine V8 compiles JavaScript directly to native machine code using just-in-time compilation before executing it instead of interpreting the JavaScript ([V8](https://v8.dev/blog/ignition-interpreter), 2016). Node.js's modules use an API designed to reduce the complexity of writing scalable server applications. It tries to do asynchronous processing on a single thread to provide more performance and scalability for applications that are supposed to handle too much web traffic ([this](https://learning.oreilly.com/library/view/web-development-with/9781785287527/ch01.html#ch01lvl2sec07), 2015).

**Bottender –** abstracts away the authentication and webhooks from each platform. It also provides API access to the **messaging-api node module** which

**NGROK -** When a user needs to access to the server with the private IP address, such request is sent to the ngrok cloud service which accepts traffic on a public address and relays that traffic through to the ngrok process running on the private IP server (in our case, Raspberry Pi)

npm -

**The messaging-api node module**

**The node-persist node module**

**EventEmitter3 node module**

**Restify & CORS node modules**

Bottender abstracts away the api authentication and webhook.

NodeJS for port

Bottender. Declarative interface with message

## Why they were used:

## Alternatives

## How they work

## How they relate to high level technologies

This information is displayed by splitting this section into each objective, then discussing (1) what the objective is and how technologies achieve it, (2) technologies that deliver said function (3) justifying the selected technology, and (4) how the function relates to higher level technologies (Networking, Web, Security, Programming Languages & libraries). This is done in order of significance for the application.

## Network Security

**What:** Authentication users are intended targets and requests cannot easily be tampered with. Three types of network security are employed to allow the backend to interface with the BCS; SSL, API tokens, and tunnelling.

* SSL
* API Tokens
* Tunnelling is exposing a local Pi server at a URL endpoint on the internet.

**How:** This application uses multiple security protocols required to develop for BCS.

* SSL: Every TLS/SSL certificate that is issued for a CA-verified entity is issued for a specific server and website domain (website address). When a person uses their browser to navigate to the address of a website with a TLS/SSL certificate, a TLS/SSL handshake (greeting) occurs between the browser and server. Information is requested from the server–which is then made visible to the person in their browser window. You will notice changes to indicate that a secure session has been initiated – for example, a trust mark will appear. If you click on the trust mark, you will see additional information such as the validity period of the TLS/SSL certificate, the domain secured, the type of TLS/SSL certificate, and the issuing CA. All of this means that a secure link is established for that session, with a unique session key, and secure communications can begin

Why:

* SSL:
  + The primary reason why SSL is used is to keep sensitive information sent across the Internet encrypted so that only the intended recipient can access it.
  + Authentication and verification
  + Data Encryption: The TLS/SSL certificate also enables encryption, which means that the sensitive information exchanged via the website cannot be intercepted and read by anyone other than the intended recipient

Options:

* HTTPS
* Keys

Selection:

BCS have strict policies on how to use.

To implement this the API Consumer must establish a URL Endpoint that can receive and process a HTTP POST request. The endpoint must also be registered with the API Publisher along with the events that the API Consumer is interested in subscribing to.

**What:**

**Why:** Facebook webhook registration requires a secure HTTPS connection ([here](https://developers.facebook.com/docs/messenger-platform/getting-started/webhook-setup/), 2020) with a domain name URL endpoint ([Aus](https://api.gov.au/standards/national_api_standards/webhooks.html), 2020). Tunnelling wraps a domain name around a local server. T a high-level, the tunneling protocol works similarly to this project. Where this project wraps a bot around a and exposes t acts in a similar way to this project. It This exposes it on the internet.

**Options:** Many tunnelling services

**Justification**:

## Listening and handling

Authentication

1. **HTTPS:** SSL encryption to protect user data and allow the server to register to PMP webhook.
2. **Tunnelling:** Tunnel to an external server to expose the local server IP as a URL endpoint on the internet.

## Tunnelling

ngrok allows you to expose a web server running on your local machine to the internet.

1. **Databasing:** Stores the state of connected users
   1. Dormant, default
   2. Waiting, if searching for partner
   3. Matched, if connected to partner
2. **Automated Responding:** Respond to messages according to user state and message text:
   1. Parrots message to matched user if connected, or
   2. Emits command to event handler where bot responds with template.

to interact with the messaging platforms any local server Since the

The tunneling interface

## Networking

LAN

WAN

Tunneling – no need to port forward

Security

Authentication

## Web

Database – state persistence

User Interface

Application server

* middleware

Webhook

Authentication

Reply

Database

Admin API

This project requires a server to handle post requests from messaging platforms. To do this the connection must be authenticated.

High level discussion of the technologies used, why were they used, alternatives, how they work (high level) and how they relate to the technologies presented in lectures.

## Technologies Used

Node

Express

Bottender

EventEmitter3

Restify

CORS

Node-persist

Natural or NLP or Python server

Gatsby client-side application

<https://neoteric.eu/blog/node-js-vs-python/#:~:text=Performance%2FSpeed,-The%20speed%20of&text=The%20faster%20it%20is%20executed,fastest%20server%2Dside%20solutions%20around>.

## Why they were used

Node is the best option for small web-based projects, whereas python is best for data science. Therefore, Node was used to host the bot, but python was used to host the NLP modelling performed asynchronously with the

Restify was used as a middleware for express to expose data to an API, making it accessible to an external application.

Bottender provides a library that abstracts much of the complexity of interfacing with webhooks from key social media providers.

EventEmitter3 was used to respond to messages asynchronously. It was used instead of the default event emitter as it is slimmer.

## Alternatives

## How they work (high-level)

## How they relate to lectures

# Design and Implementation (5 pages)

Discuss details of the design and implementation, how it works, challenges you faced, configuration, experimentation, results etc. Include a system diagram and discuss future directions / improvements

## Details

## System Diagram

## How it works

## Challenges

## Configuration

## Experimentation

## Results

## Future Directions / Improvements

# References

Idea:

A messenger bot that helps you connect to people and information.

Networking & The Web

Networking: HTTP & WebSocket connection to server that is hosted on the pi. Used to edit facts about the server without restarting it.

The Web: NodeJS server that interfaces with different APIs.

Features:

1. Omegle like anonymous chat service: connect lonely strangers online
   1. /start triggers this feature
   2. If another user is waiting, users connect
   3. Else the user status is changed to waiting
2. Subscription to information that matters to the user
   1. Online game deals
   2. (maybe)
      1. Betting odds
      2. Financial market
      3. Quotes
      4. News
3. Webserver displaying information about the application that can interact with he
4. (if another thing is needed) A LED turns on for 500ms when a user triggers an event on the server

Technologies:

1. Server: NodeJS
2. Anonymous messaging done by JS on node server.
3. Hardware interface: onoff
4. Bot interface: Bottender
5. Gaming deals:
   1. Cheapshark: <https://www.cheapshark.com/api/1.0/deals>
   2. (optionally) Match with IGDB for more game information: <https://api.igdb.com/v4/games>
6. Betting odds: The Odds Api
7. Quotes: <http://quotes.rest/qod.json?category=inspire>
8. News API: <http://newsapi.org/v2/top-headlines>

Checkpoint 1:

Omegle like connection to stranger

Server API to gaming deals. User preferences are stored on node server which filter the broadcast signal to all subscribed users.

Checkpoint 2:

Websocket to server that renders to a HTML endpoint which enables the user to modify facts about the server.