Second Prototype Test Plan – 02/29/2024

To: Professor Pisano

Team: 7 (MuseumMate)

Date: 02/29/2024

Group Members:

John Culley jculley@bu.edu

Kai Imery kimery@bu.edu

Yanyyang Zhang yz21@bu.edu

Ananth Sanjay ananths@bu.edu

Kwadwo Osafo kosafo@bu.edu

1.0 Required Materials

Hardware:

- **Makerfabs ESP32 UWB**
- **Adafruit Li-Ion/Li-Poly Charger [v1.2]**
- **3D Printed Enclosure for Beacons**
- **PCB for Handheld Device**
- **USB 5V Power Supply**
- **PKCELL LP552535 3.7V 420mAh**
- **Passive RFID Tags**
- **RC522 RFID Scanner Module**

Software:

Server:

- **Node.js**
- **Express.js**
- **InfluxDB**
- **MinIO**

ChatGPT Server:

- **Gin Framework**
- **ChatGPT API (3.5 Turbo)**
- **Golang**

Frontend:

- **React Native**
- **Expo**

- **React**

User Device:

- **UWB Module**
- **Battery Charging System**
- **RFID Reader**
- **Connection to Campus WiFi**
- **UDP Client**

Beacon:

- **UWB Module**

2.0 Prototype Architecture

The prototype we are demonstrating can be split between two broad categories; hardware and software. The software consists of separate applications that run the server (backend), frontend, handheld user devices (TourTags), and beacons. The backend consists of various code that describes the functions of our networking and storage capabilities. Our handheld devices (TourTags) send data to a server on our backend through UDP and the data received is stored in an InfluxDB database through the use of an API. The server is also capable of using trilateration to determine what room a given TourTag is in, given the nearest three beacons. The front end consists of a React.js application which allows users to log into our system and receive positioning data corresponding to their respective TourTags. It also includes several quality-of-life features, which include, but are not limited to, text-to-speech features in multiple languages, multimedia overlays giving additional information about specific exhibits, and a dashboard for system administrators. The last part of our software consists of code that allows the TourTags to detect UWB signals between themselves and the beacons, which allows them to compute an accurate measurement of the distance between themselves and the beacons. This data is then processed and sent to the server through the use of WiFi at recurring intervals.

Our hardware has shifted significantly since our first prototype test, mainly due to the replacement of Adafruit Huzzah32 ESP32 Feather Boards with Makerfabs ESP32 UWB, which was done in order to utilize UWB technology to determine distances between TourTags and the beacons, which we deduced will be more robust and accurate than BLE technology. Additionally, we have included a Li-Ion Charger in the TourTag design, which allows for charging of the handheld device without needing to detach the lithium-ion battery from the device. The ESP32 UWB is split to serve as either a TourTag or a beacon (housed in a 3D-printed enclosure). The beacons are powered by USB 5V power supply units connected directly to electrical sockets, and the TourTags are powered by PKCELL LP552535, which is a rechargeable lithium-ion mobile power supply unit. RFID modules will also be attached to the TourTags to demonstrate their functionality.

3.0 Setup Procedure

Pre-testing Setup Procedure:

Physical Setup:

1) Plug in Beacons into wall sockets using 5V power supply units in designated locations shown in **Figure 1** ensuring that the ESP is facing the right way to allow the most precise positioning
2) Plug PKCELL power supply unit into User Device

Software Setup:

1) Run the command **node index.js** to start backend
2) Run command **npx expo start** to start frontend
3) Run command **go main.go** to start ChatGPT Server

4.0 Testing Procedure

1) Render app
2) Run Node.js Server
3) Scan QR code on TourTag enclosure
4) Follow Screens Navigation
5) Display Current Location of User Device from HTTP query

5.0 Measurable Criteria

Hardware:

1. Handheld (User) devices accurately deduce the distance between themselves and a given beacon within approximately +/- 0.2m error bounds.
2. 3D printed enclosures should not interfere with the WiFi and UWB antennas of the enclosed beacon
3. RFID tag is successfully scanned by handheld device, and the corresponding RFID tag ID is recorded.
4. Updated distance measurements from UWB readings should be sent to the central server upon the detection of signals from 3 unique beacons in a single sampling interval.
5. Contents of packets sent via UDP to the central server should be consistent with predetermined protocols.

Server:

1. Successfully receives UWB data via UDP from user device
2. Stores and processes UWB data with a queue that is updated each time new data is received
3. API Testing:
    a. /location/:userID (GET)
        i. Returns user location to the front end
    b. /path/:userID (GET)
        i. Given a userID and room, return route to get to destination
    c. /tsp-path (POST)

        i. Given an array of rooms, return the shortest path to visit all the rooms
    d. /rfid/:bucketName (GET)
        i. Given a bucket name, return all associated multimedia from minIO
4. In 5-second intervals, occupancy by room is obtained and sent to InfluxDB to be timestamped

ChatGPT Server:

1. API Testing:
    a. /chat (POST)
        i. Returns the response from ChatGPT based on the prompt given in the request
        ii. Upon successful request, response is either retrieved from cache or if it is a new prompt it is generated with the ChatGPT API and returned

Front-End:

1. Initializes the app and swiftly navigates to the HomeScreen within 2 seconds, offering primary navigation.
2. Activates the BarcodeScanner to utilize the device's camera for scanning a barcode and displaying results within 5 seconds.
3. Displays a list of available tours, including TimedTour and Explore, within 3 seconds upon selecting TourTypes.
4. Accurately locates the user's device and retrieves the corresponding map image within 6 seconds using the CurrentLocation feature.
5. Ensures smooth screen transitions without freezes exceeding 2 seconds and responsive interactive elements within 2 seconds.
6. API Testing:
    a. /location/:userID (GET)
        i. Retrieves the current location of the user's device and updates the display or UI elements.
    b. /path/:userID (GET)
        i. Provides navigation route for a specified userID to a selected room within the app.
    c. /tsp-path (POST)
        i. Receives a user's selected rooms and computes the shortest visitation path.
    d. /rfid/:bucketName (GET)
        i. Returns all multimedia associated with a given bucket name from minIO.
    e. WebSocket ws://10.192.45.20:8080
        i. Establishes real-time connection for RFID data, updating UI with new object info.
    f. Google Translate API https://translation.googleapis.com/language/translate/v2
        i. Translates text to selected user language for improved accessibility.
    g. GoLang API http://10.192.39.193:4040/chat (POST)

i. Accepts prompts and returns informative responses for interactive user engagement.

Figure 1: Locations of beacons and room sections