Android Apps

Smartphone

General path: "*your folder*\Smartphone\Smartphone_App_Source\app\src\main\java\com\example\srikate\ibeacondemo"

\fragements\QRFragment.java:

- Callback
- BarcodeDetector
- Sends intent to WebView

\fragements\WebViewTestFragment.java:

- Display URL from QR-code

\model\BeaconDeviceModel.java:

- minor, major, signal, average, smallest, largest, mostFrequent
- contains functions to retrieve beacons
- sorting of list
- function for updating processed data variables
- most frequent helper function

\utils\JavaScriptInterface.java:

- Contains functions that can be called from JavaScript on a web-server

\utils\UiHelper.java:

- Does a lot of UI things for us
- error messages
- pop up dialogue for permission

\MainActivity.java:

- Loads UI
- Sets up Bluetooth
- Starts on QR-Fragment
- Functionality for switching between fragments
- Initializes bluetooth Adapter/BLE

- Starts bluetooth scanning
- findBeaconPattern() //Filters advertisements such that only iBeacon advertisements gets caught
- foundBeacon() //
- updateBeacon()
- lots of beacon helper functions
- Can find all retrieval methods in mainActivity class.

Tablet

General path: "*your folder*\Tablet\Tablet_App_Source\app\src\main\java\com\example\srikate\ibeacondemo"

\fragments\BeaconSimlatorFragment.java:

- SetupBeacon()
- Sets up UI for beacon configurations
- initial screen for tablet

\fragments\WebViewTestFragment.java:

- hosts the website

\utils\UIHelper:

- Does a lot of UI things for us
- error messages
- pop up dialogue for permission

\MainActivity.java:

- Sets up UI
- Sets up Bluetooth
- functionality for swapping webview
- transmitIBeacon() //Starts Bluetooth advertisements
- bluetooth helper methods
- Checks bluetooth permission

WebApp

General path: "*your folder*"\Bachelors_Project\WebApp\BachelorsProjectWebApp

\Controllers\HomeController:

- Controller for switching between views. This allows for setting different parameters in the ViewBag to be used in the different views of the web app.
- Also contains database access point through DBMapper class.

Smartphone Interface

\Controllers\HomeControler, Navigation():

- Routing to navigation, takes input string, which contains target beacon and floor plan ID.
- Retrieves relevant data from the database on the floor plan ID and loads the view with this info.

\Views\Home\Navigation:

- Loads view and background from ViewBag
- background.onload:
    - Defines the canvas which is used to draw the floor plan using calculateOffset and convertObejctsToCoordinates.
- DrawVisuals:
    - Renders the background and calls drawTrackCircles.
- drawTrackCircles:

        - Makes a second canvas and draws all the tracks collected from the database for the given floor plan.
        - Also runs the checks if the destination has been reached.
    - retrieveBestBeacon:
        - Calls Android through the JavaScript bridge to get the ID of the best beacon, which is found in the local list and passed to drawVisuals.

Tablet Interface

\Views\Home\Index:

- Main screen for tablet displays. Contains three segments: Profile Card, Calendar, and Research Pie and Search bar

\Views\Home\FindOffice:

- Loads datatable based on input search from ViewBag.
- Makes all rows in datatable clickable with QR code generation based on attributes in data. Passes QR code and some more attributes to pop-up.

Floor plan tool

\Controllers\HomeController, UploadFloorPlan():

- Gives the possibility to upload a URL as your floor plan, which you then can name. When you hit the button, it'll send the info to the database. It then redirects to the EditPlan with the new floor plan ID.

\Controllers\HomeController, EditPlan():

- Loads the floor plan based on the ID, including offices. (still needs to be edited directly in database, ups)
- When done with editing and save is clicked, it splits all of the strings defined in the BeaconModel parameter into beacons and tracks, which are matched with their positions and saved to the database.

\Views\Home\EditPlan:

- Starts by loading two canvases, one for the background and one for drawing the beacons and tracks.
- toggleBeacon() and toggleTrack() are helper methods for making sure that you don't place beacons or tracks on the buttons.
- scalePixels() converts screen clicks to clicks within the picture.
- Positions of beacons and tracks are StringSerialized for parameter passing. The same goes for beacon IDs and office IDs.
- save() takes all the data and puts it into a model, which is simply a Data Transfer Object. It's then posted, which takes it to the HomeContoller EditPlan Post.
- printMousePos() handles the general onClick. This will create a beacon or track, if the corresponding button has been previously pressed.
- reset() resets all local variables and clears the second canvas, i.e. all the beacons and tracks places so far.