0.1 Scheduled Attendance Taking Services

The attendance taking functionality will be implemented as Android background services that are scheduled to run at the time the session will start. Four different services will be implemented for each of the following functions: opening attendance, closing attendance, advertising attendance tokens, and scanning for devices advertising attendance tokens. The first two services will only run on a device with lecturer credentials. On app startup, the services will be scheduled by an Android AlarmManager.

0.2 Custom GATT Profile

A custom GATT profile (dubbed Beam Profile) will be utilised for data transfer between two devices using the app. The Beam Profile contains one custom GATT service (Beam Service) which contains one characteristic, known as Attendance Token, that holds a string value generated as the hash of a lecture session ID. The value of this characteristic can be read by the GATT client, written by the GATT server, and the GATT client can be notified of changes in the characteristic.

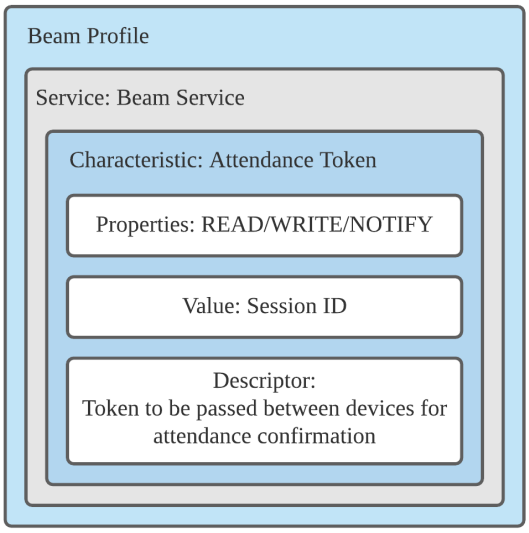


Figure 0.2.1: *Custom GATT Profile*

Regarding the peripheral role in the app, the lecturer's device acts as the first peripheral device. The session ID will be written into the service characteristic. The device will then advertise the Beam Service UUID and wait for connection request from a central device. Once a connection is established, the peripheral acts as the GATT server. Whenever the central device (GATT client) requests to read the value of the characteristic, the server will send a response containing the value. This is how the attendance token is passed between devices.

Regarding the central role in the app, the students' devices are central devices by default. Once the session has started (based on OS time), the central device will begin scanning for devices that advertise the Beam Service UUID. Once it detects one, it'll initiate a connection and begin acting as the GATT client. The device will immediately request to read the value of the characteristic within the service. Upon receiving the attendance token, the value is compared to that in the database. Once attendance has been taken, the device closes the connection and switches to a peripheral role.

0.3 Database and Web Hosting Choice

The database used by the app will be Firebase Realtime Database, which is a cloud hosted database whereby data is stored as a JSON tree. All clients share one Realtime Database instance. The data stored by the app is estimated to take up a few GBs of space and only basic querying is required by the app. Additionally, only one instance of the database is needed for all clients. Thus, Firebase recommended Realtime Database as a better choice compared to Firebase Cloud Firestore, a NoSQL relational database, which is better suited for apps that require multiple databases, advanced querying, and hundreds of GBs to TBs of space. Firebase was chosen over other web services (such as Amazon Web Services) because it's hosting services are free (to a certain degree) and the API is estimated to be easier to learn and work with.

The website is hosted using Firebase Hosting: `https://beam-5845a.web.app/`. Email for authentication is admin@nottingham.edu.my and the password is *password*.

0.4 Firebase Authentication Design

There are three types of user accounts: student, lecturer, and admin. Student and lecturer accounts grant permission to access the application, while admin accounts grant access to the administration website. Each student and lecturer accounts will have their own User ID stored in Firebase Authentication. The User ID will be used to query the database to search to student or lecturer details. The admin accounts are created on the console by the team and cannot be created in other way.

0.5 User Interface Navigation

a. Website

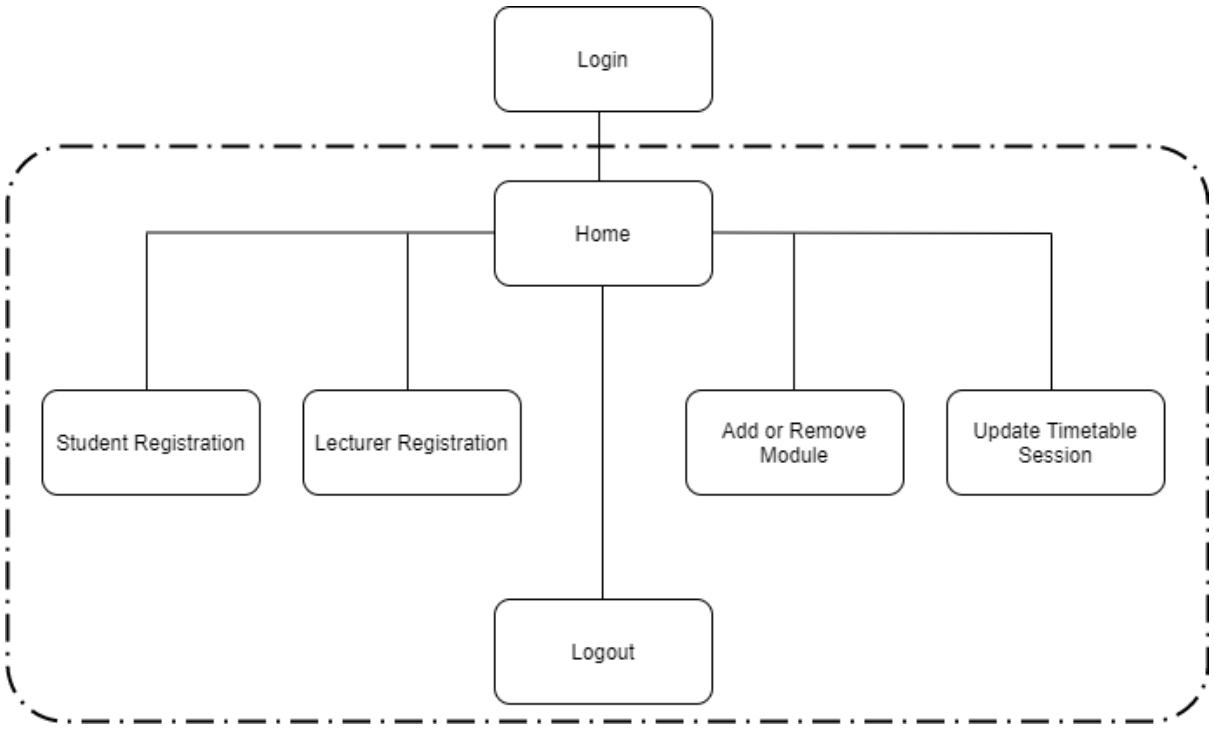
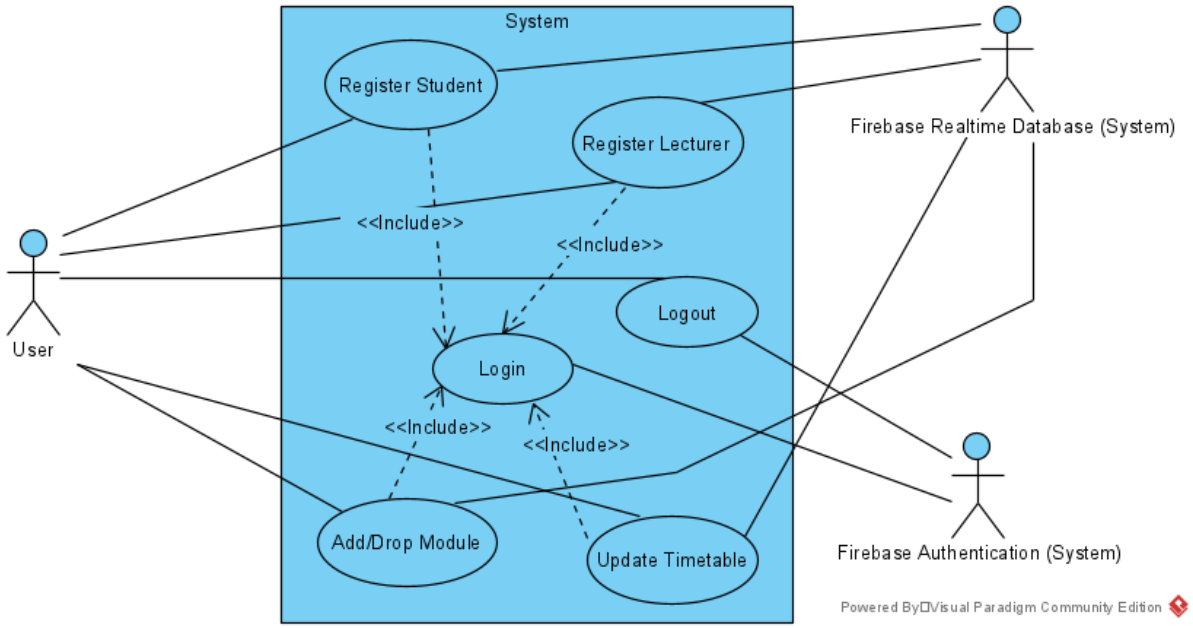


Figure 0.5.1: *Sitemap of administration website. Components inside the dotted box belong to a single-page application*

b. Application

0.6 Use Case Diagrams

a. Website



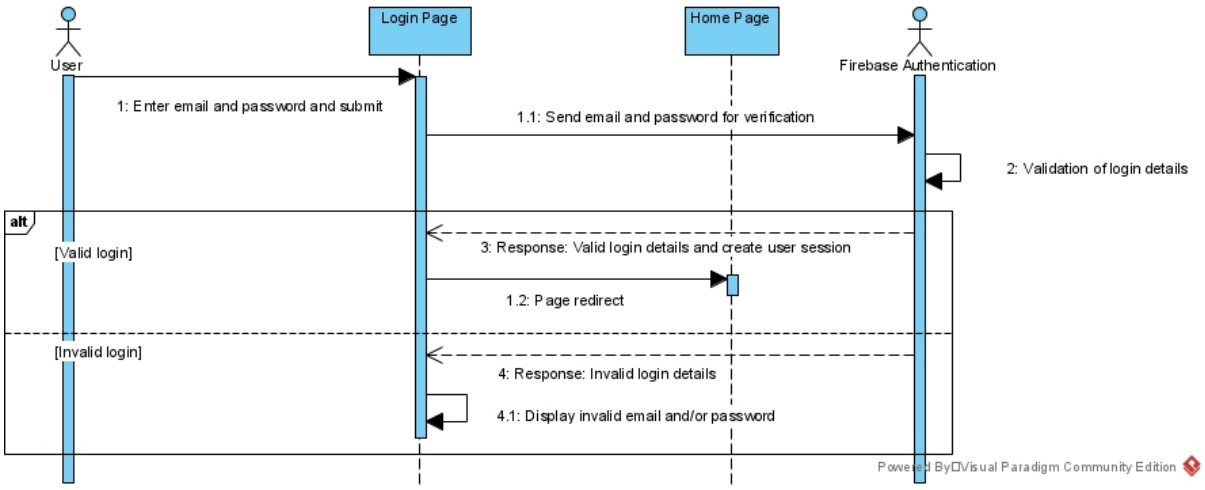
b. Application

TODO

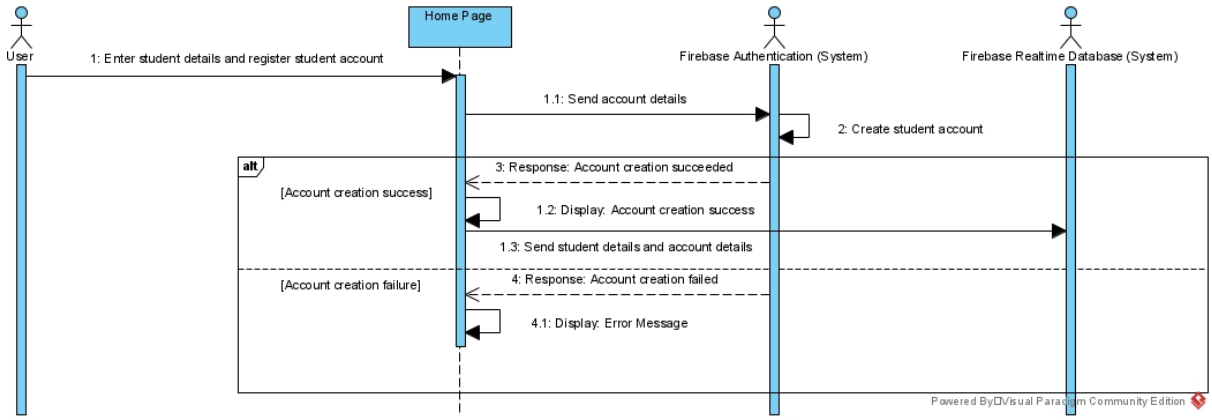
0.7 Sequence Diagrams

a. Website

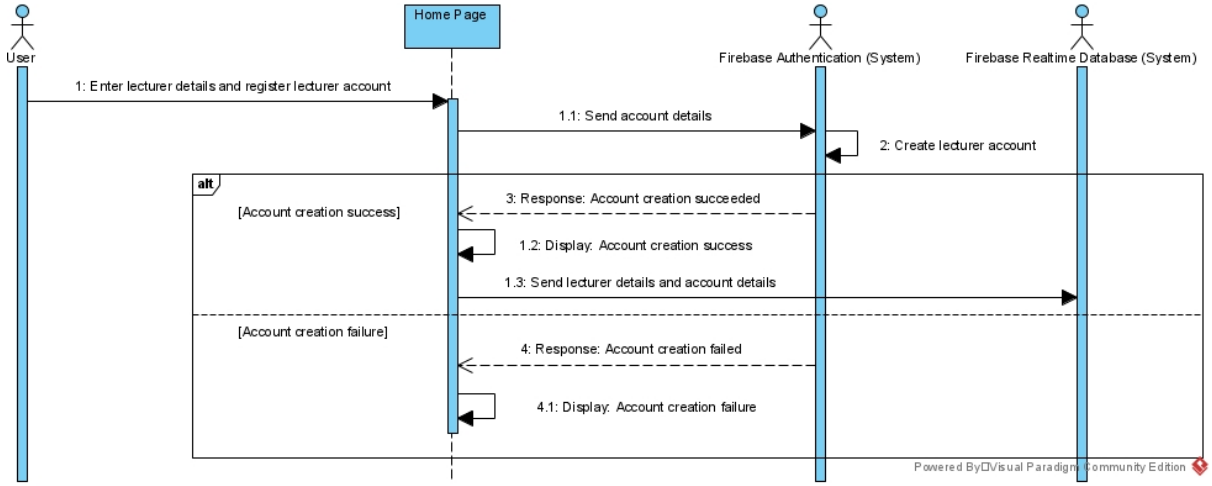
Login



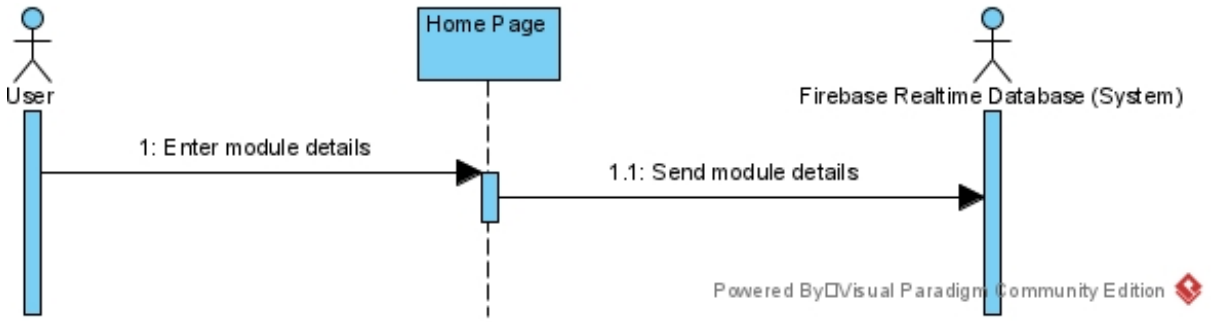
Student Registration



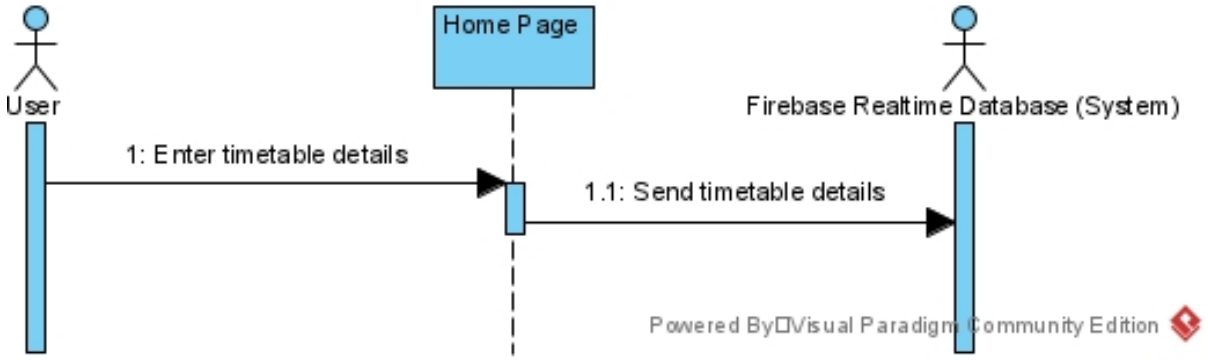
Lecturer Registration



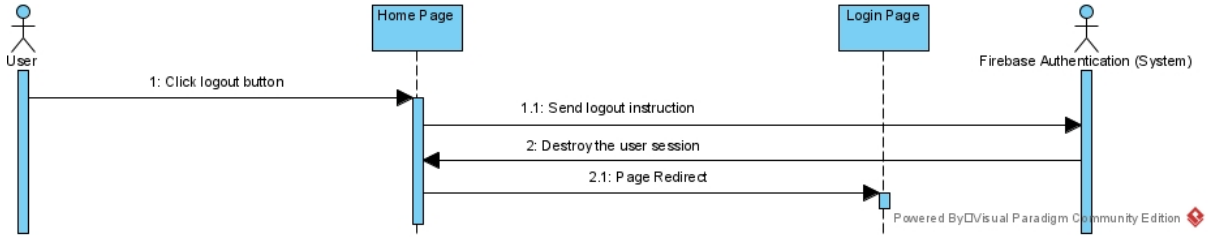
Update Module Information



Update Timetable

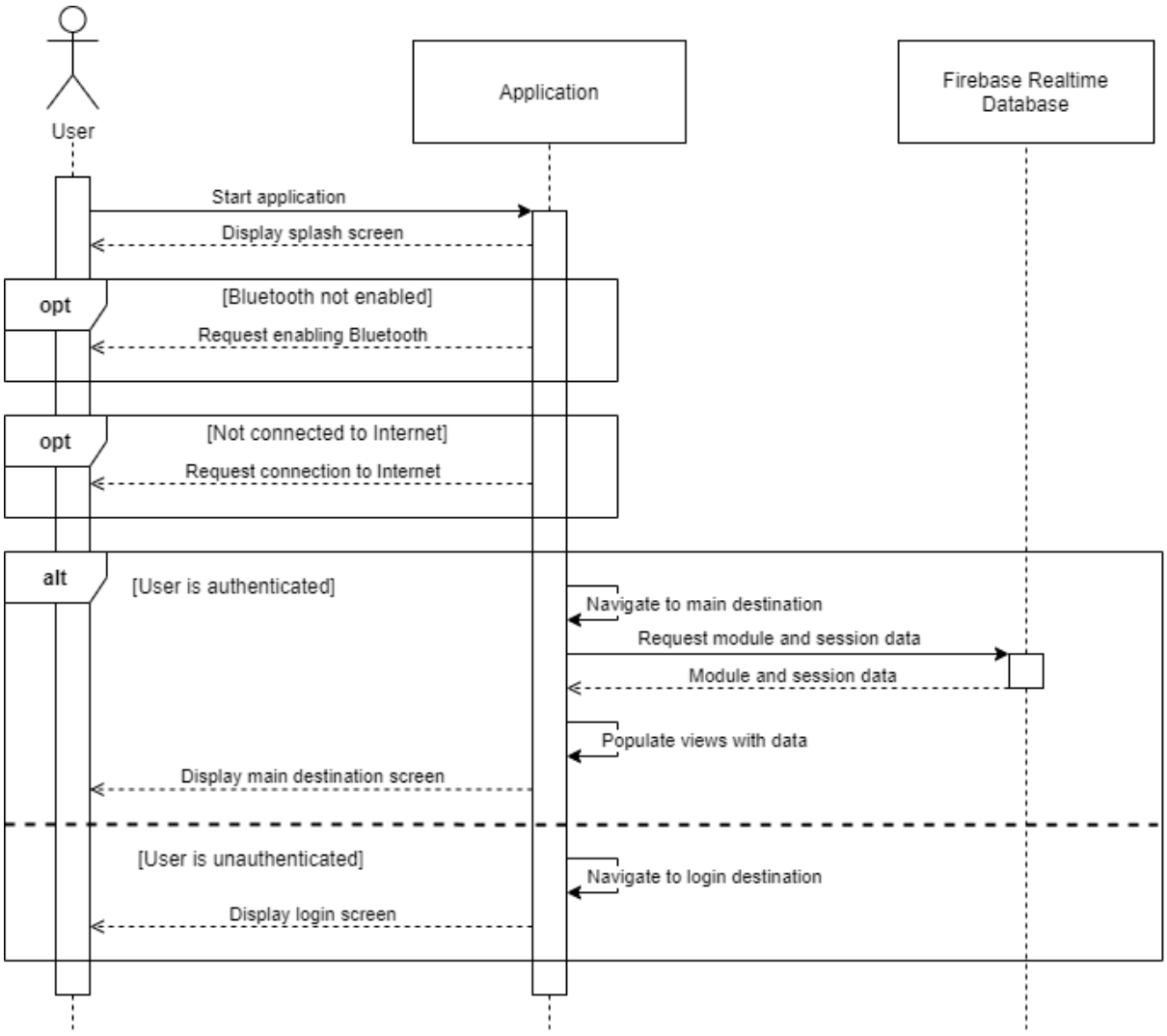


Logout

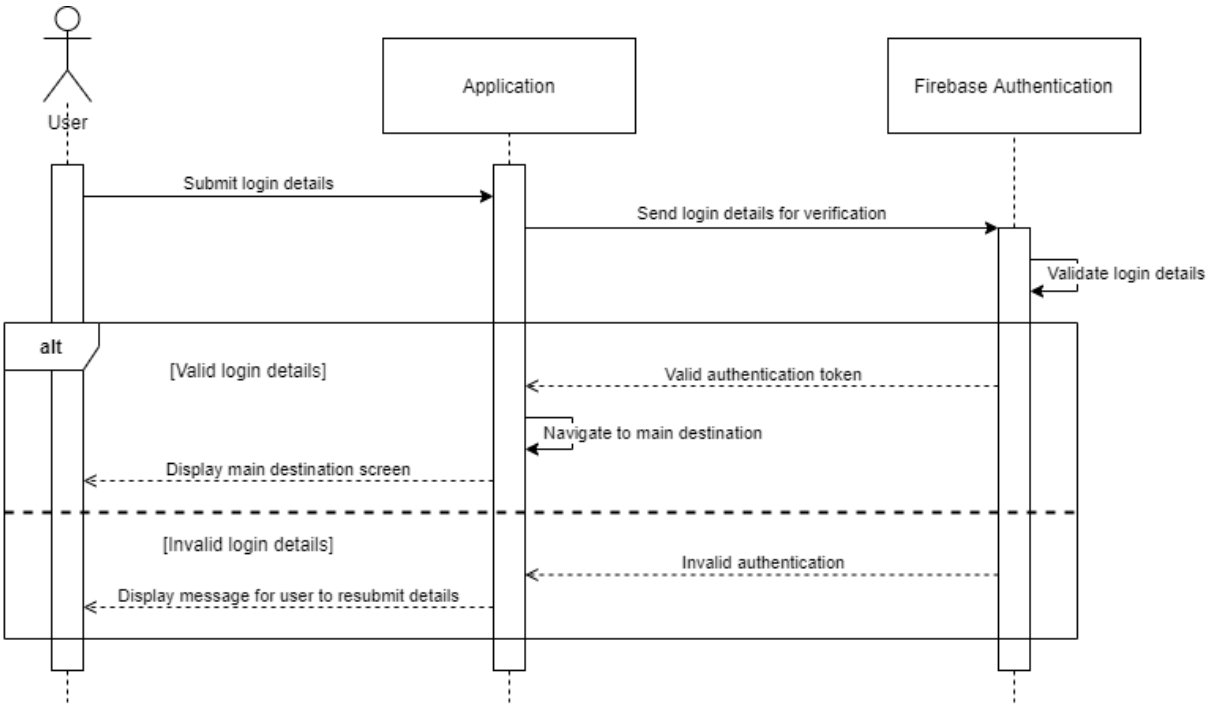


b. Application

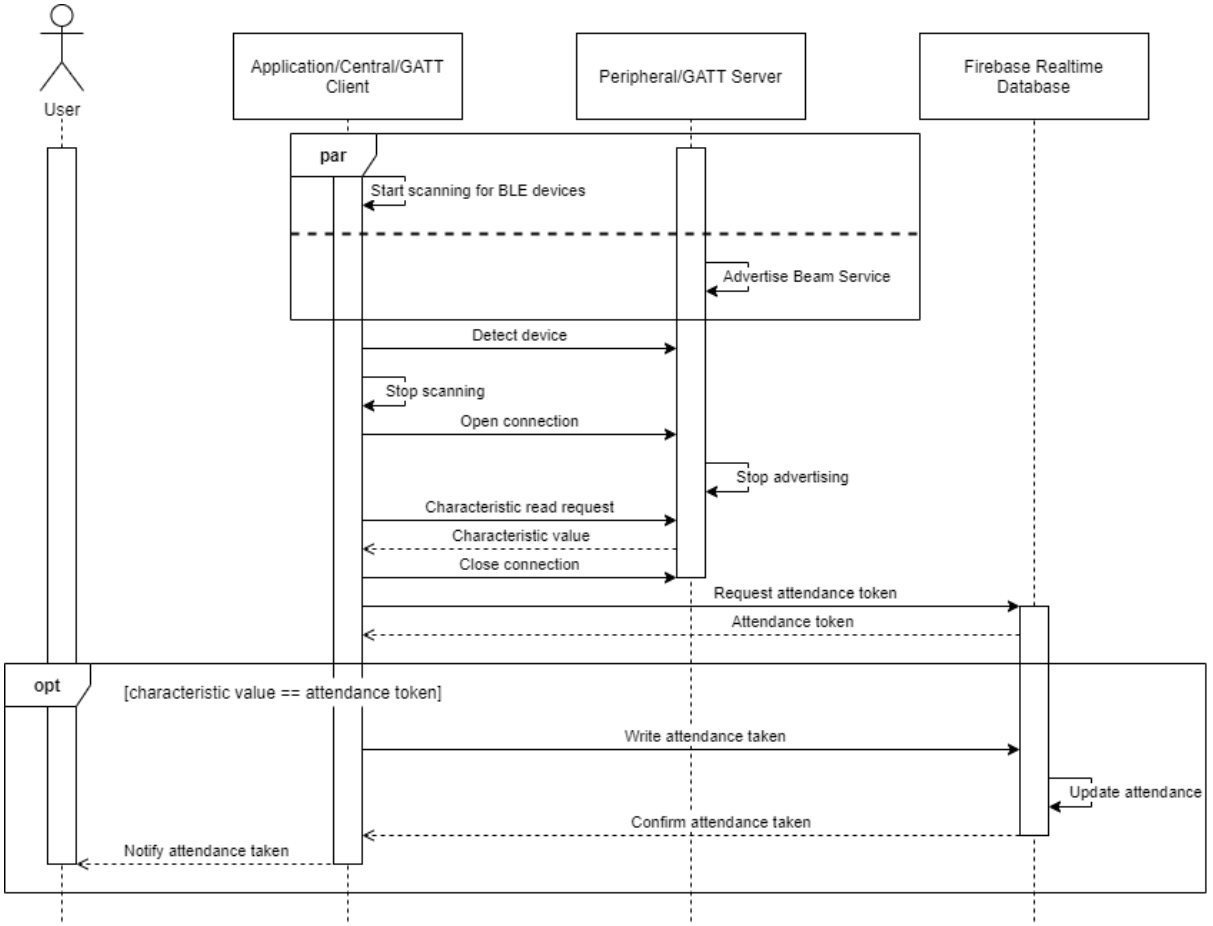
i. Application Start


User
Application
Firebase Realtime Database
Start application
Display splash screen
opt
[Bluetooth not enabled]
Request enabling Bluetooth
opt
[Not connected to Internet]
Request connection to Internet
alt
[User is authenticated]
Navigate to main destination
Request module and session data
Module and session data
Populate views with data
Display main destination screen
[User is unauthenticated]
Navigate to login destination
Display login screen

ii. Login



iii. Attendance Taking



0.8 Firebase Realtime Database Design

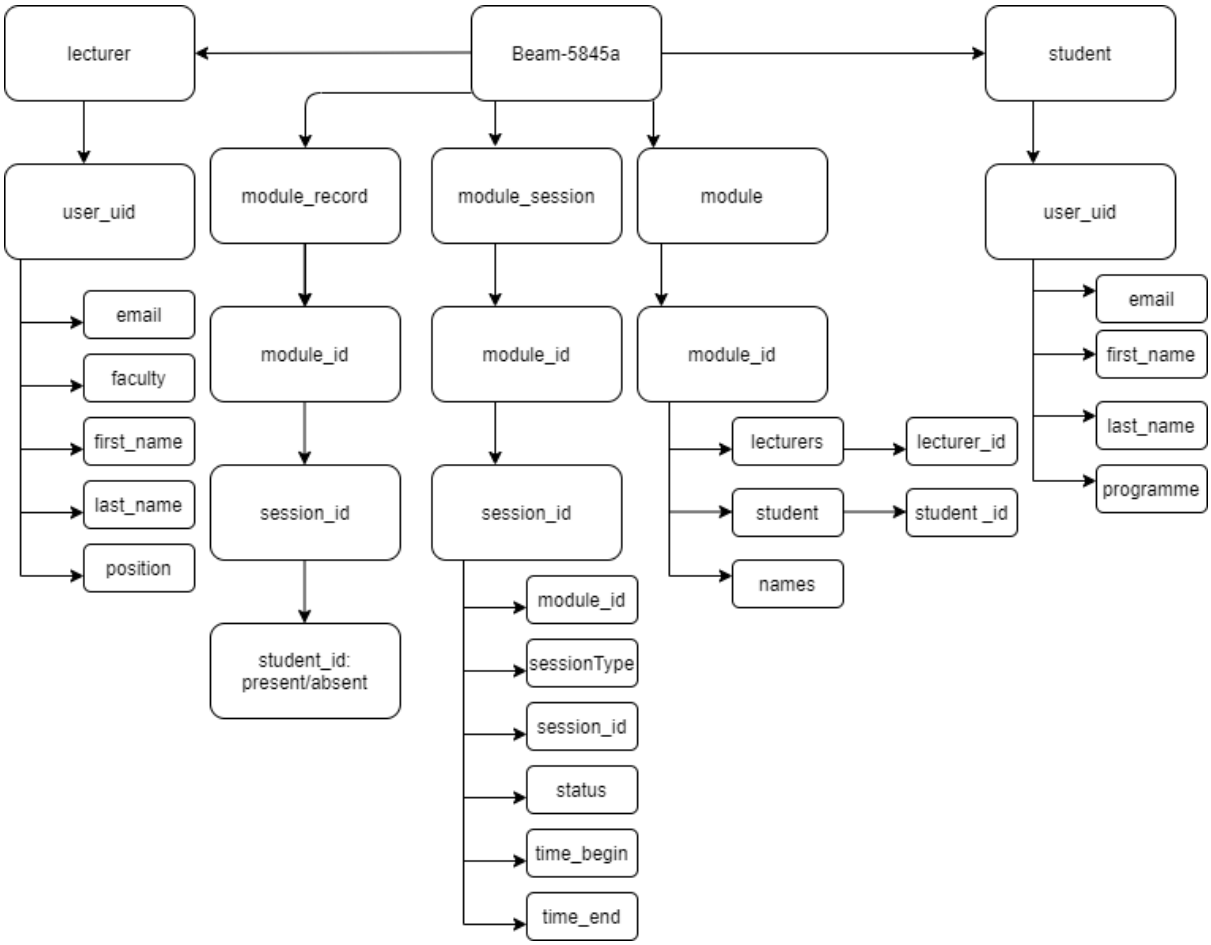


Figure 0.8.1: *Database design part 1*

The *lecturer* node groups the lecturer details by *user_uid*, which will be fetched to populate the profile in the app. The *module_record* node stores the students who attended an academic session by *module_id*. The app will calculate the percentage of students who attended the session. The *module_session* node groups each session by their module. The *module* node groups module details by *module_id*. The *student* node groups the student details by *user_uid*, which will be fetched to populate the profile in the app.

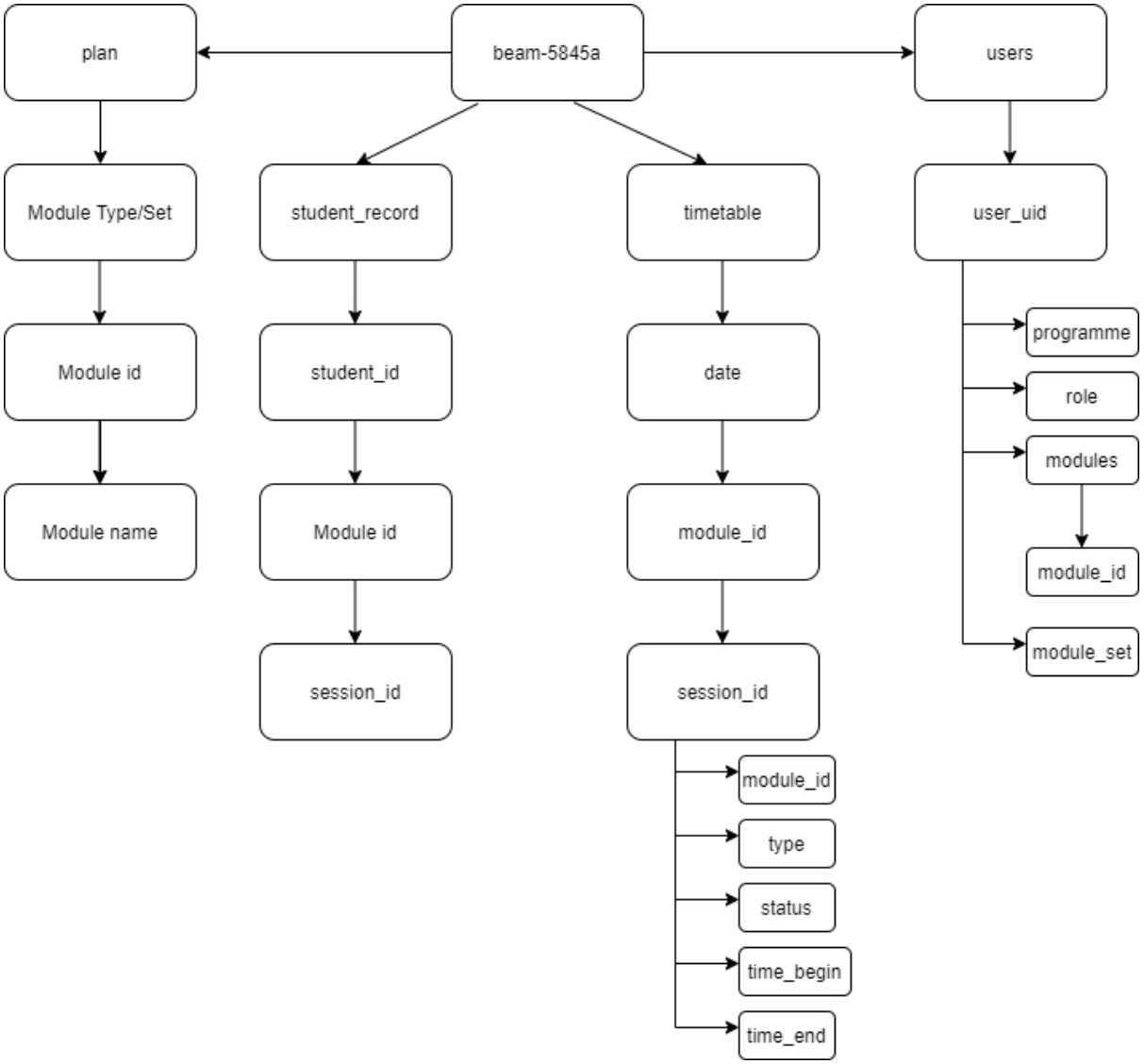


Figure 0.8.2: *Database design part 2*

The *plan* node stores the modules by programme. Modules in a programme are group by module type or set. Module type such as Core and Elective are applicable for students, while module set groups the modules taught by a lecturer. Grouping modules in this way allows the admin to perform only one data entry (by type or set) instead of entering each module one-by-one. The *student_record* node stores all attended academic sessions of a student, grouped by module id. The *timetable* node stores each academic session and its module id for every date when there is an academic session. The *user* node stores important identification details. The *programme* node groups all the modules a student is taking in an academic plan. The *role* node identifies whether a user is a student or lecturer. The *module* set groups all the modules a lecturer is teaching.