**Finger Band**

**Team4**

Uiling

Hsuanchen

Carson

楊皓崴

Judy

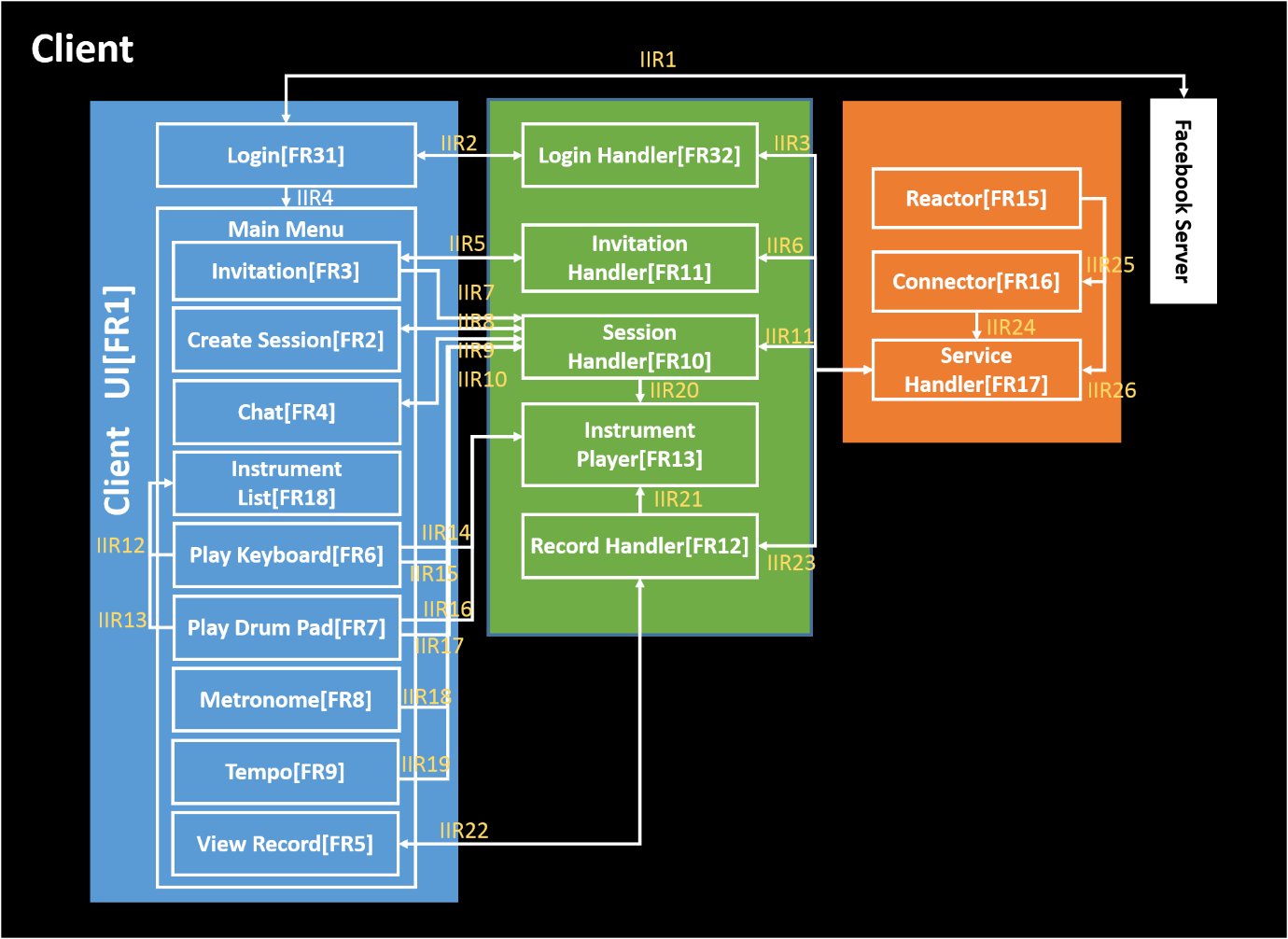
Cliff

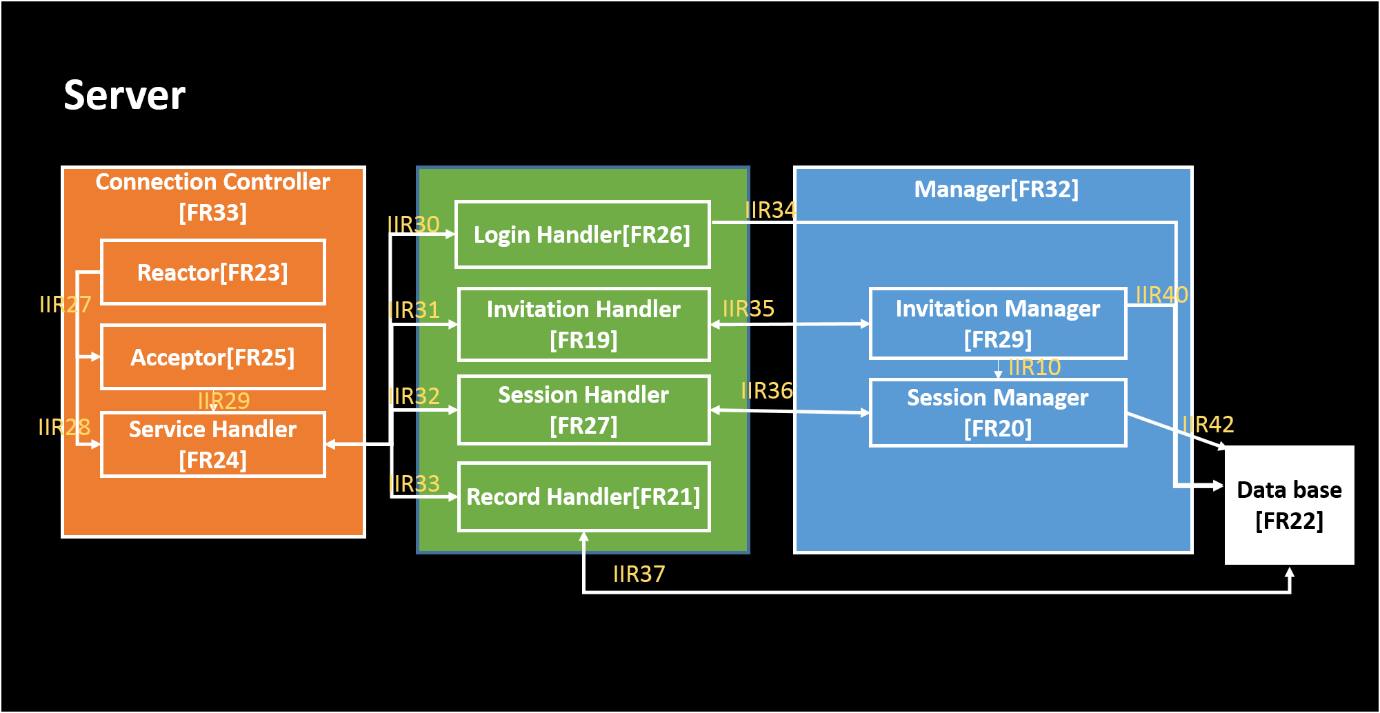
* **Introduction**

Jam Session (即席演奏會) 是一種音樂創作者彼此之間透過共同演奏，相互激勵並藉其誘發即時創作靈感的一種團練儀式。傳統認知上的 Jam Session 的事前準備非常麻煩，團員除了需要自行攜帶樂器 (爵士鼓、keyboard或低音提琴等大型樂器尤其麻煩) 之外，通常需要事先向錄音室承租場地 (一筆不小的開銷)。若自行找場地 (通常是廢棄無人的密閉式空間)，還必須自行解決電源問題，並同時準備錄音設備、擴音器、混音器等器材，是一件非常麻煩的一件事，因而使多數音樂玩家 (在成本考量下) 被迫減少了藉由 Jam Session 所促成的互動式創作活動，非常可惜。

Finger Band 是一個專為音樂愛好與創作者的 mobile service，嘗試替音樂玩家 (尤其是學生) 解決上述 Jam Session 會面臨的經濟壓力與樂器搬運問題，提供了使用者一個即時創作、攜帶方便又省時省錢的即席演奏環境 (jam session environment)。Finger Band 同時借用了 Facebook 的 Messenger 與 AOUTH 認證機制 提供使用者相關的會員認證管理與即時訊息推播等服務。

* **System Architecture**





* **Requirement**
  + **Functional Requirement**

|  |  |
| --- | --- |
| Client: |  |
| FR1 | **Client UI** |
|  | There are two activities.  Activity 1 = Login Screen  Activity 2 = Main Screen  Main Screen is divided into two areas, Button Area and Content Area.  Each button has a corresponding content. The relationship between contents and buttons are described below:  Button Area contains the following buttons:   * Create Session * Invitation * Session * Record * Key board |
| FR2 | **Create Session** |
|  | Press this button to create a session. If the session is successfully created, this button will be replaced by a [InviteFriends] button at the same location. |
| FR3 | **Invitation** |
|  | Press this button to display the <invitationUIFragment>, which contains a list of the user’s Facebook friends who has already downloaded our app. This button will not appear under single-player mode. |
| FR4 | **Chat** |
|  | Press this button to display the <chatUIFragment> in the Content Area, which contains a message input text field, a submit button, and chat history. This button will not appear under single-player mode. |
| FR5 | **View Record** |
|  | Press this button to display the <recordListUIFragment> in the Content Area, which contains a list of records from sessions that the user has participated. |
| FR6 | **Play Keyboard** |
|  | Press this button to display the <keyboardUIFragment> in the Content Area, which contains a list of instrument (along with checkboxes and a submit button) that the user can play with the keyboard. For ease of identification, this list is divided into 4 segments: 1) KeyInstrument, WindInstrument, BrassInstrument, 4) StringInstrument. Only one instrument can be selected at once. After the instrument is selected, a keyboard consists of 88 keys will appear in the Content Area. Each key is represented a button on the screen. If pressed, a particular sound of the selected instrument will be produced. At the same time, each button press will also generate a set of data (i.e. channel, program, volume, duration, notes) to be sent to the server and be rebroadcasted immediately by the server to the rest of the session member. |
| FR7 | **Play DrumPad (to do)** |
|  | Press this button to display the <drumPadUIFragment> in the Content Area, which contains a list of drums and sound effects (along with checkboxes and a submit button) that the user can play with the drum pad. For ease of identification, this list is divided into 5 segments: 1) FixedDrumKit, 2) FixedPercussion, 3) FixedSoundEffects, 4) ChromaticPercussion, 5) ChromaticSoundEffects. A maximum of 12 drums and/or effects can be selected at once. After the selection is made, a drum pad consists of 12 buttons will appear. If pressed, each button will produce the sound of its corresponding drum or sound effect. At the same time, each button press will also generate a set of data (i.e. channel, program, volume, duration, notes) to be sent to the server and be rebroadcasted immediately by the server to the rest of the session member. Certain drums or sound effects are chromatic, meaning that they can be set to play at a different pitch (sound wave frequency) within a fixed range. For these drums or sound effects, if their corresponding button are “long-pressed” (press and hold over 1 second), then a pitch adjustment widget will appear for the user to make the proper adjustment. Buttons for all chromatic drums and effects should be “marked” in order for the user to tell the difference. |
| FR8 | **Metronome(To do)** |
|  | Press this button to turn the sound of the metronome to be on or off. |
| FR9 | **Tempo(To do)** |
|  | Press this button to increase/decrease the tempo of the metronome. This button is only activated for the session creator. This button will appear as a deactivated button for other session member. |
| FR10 | **Session Handler** |
|  | After login, the user can create a Session.  If user has created a session and is in it, one can set the tempo.  After the creator invite friends, one could accept the invitation.  If a member not creator leave, session will no longer send a sound to him/her neither send a sound to others.  If creator leaves, the session will be ended, and every member will be kicked. |
| FR11 | **Invitation Handler** |
|  | Once one click invite button, it will use Invitation Manager to show friend list.  Invitation Manager can get friend list from server via Invitation Handler.  User use check box to decide who is invited. After User decide the invitee list, Invitation Manager will send it to Server via Invitation Handler.  Friend in friend list contain one's name, portrait Url.  Once restart the invitation page, Invitation Manager will get Invitation list from server via Invitation Handler.  Invitation contains creator and session port number.  User can accept or reject to an Invitation.  Once User reject an invitation, Invitation will be delete.  Once User accept an invitation, it will go to join a session. |
| FR12 | **Record Handler** |
|  | Once restart the record page, Record Handler will get record list from server.And it will show just simple information of record, for example, record play time, record create time, with who.....etc.  User can click Record to get MidiFile from Server via Record Handler then Application will play the record by Instrument Player.  MidiFIle contain timesline of which note of which instrument is played. |
| FR13 | **Instrument Player** |
|  | Instrument Player will receive Instrument and note number to play the certain note and sound by adjust channel, program and note on/off with javax.sound. |
| FR15 | **Reactor** |
|  | The Reactor will monitor the connection of EventHandler (Connector, ServiceHandler). If there is an event occurred, it will call the handle method of corresponding EventHandler to handle event. |
| FR16 | **Connector** |
|  | The Connector is responsible for establishing the connection. It will make sure the connection is completed by handling the ISCONNECTABLE event. And then it delegate the connection to ServiceHandler attached to it. |
| FR17 | **Service** **Handler** |
|  | SessionIDTask will tell the user the sessionID once this user are in the session.  SessionOverTask will tell the user to leave the session once the creator has ended the session  TransmitMessageTask will receive a message once the other user In the session send a message  TransmitMusicTask will receive a music once the other user in the session play a sound. |
| FR18 | **Instrument** **list** |
|  | The Instrument list will list the instruments user can play. By default, it is piano. (We will extend more instruments in the future) |
| FR31 | **Login** |
|  | Login Module will access the facebook account. It can receives the information of acknowledgment from facebook server.  If user login with facebook though other application, then android manager will catch the information of facebook automatically. |
| FR32 | **Login Handler** |
|  | Login Handler Module will receive the information of facebook login result though Login Module to handle. If handle successfully, then it will get facebook id and name. |

**Server:**

|  |  |
| --- | --- |
|  |  |
| FR19 | **Invitation Handler** |
|  | **Initiate an invitation**  User can initiate an invitation in the system.  Server will create an invitation with the port number and user's account. When the invitation is initiated, sever (Invitation Manager) will record the invitation.  **Request friend list**  Server (Invitation Manager) can request user's friends from Facebook server.  **Check App availability**  Server (Invitation Manager) can check if one user has installed the App or not  **Record Invitation**  Server (Invitation Manger) holds all the invitation information  **Return User's Invitation**  Server will return users' own Invitation list  **Remove an invitation**  Once user accept or reject an invitation, sever will remove the invitation from user's own list  **Respond to user's acceptance**  After server remove the invitation from user's own list, server will send the session port back to user |
| FR20 | **Session Manager** |
|  | Once the session has started, this will be created and add the creator to the session.  Once other user join the session, this session manager will create a trackrecord(if this user has not joined before) for the user.  When receive a music and message, this will tell all the other member in the session.  When ended, this session will notify all member in the session. |
| FR26 | **Login Handler** |
|  | Login handler can receive user information from client. It will send information to database to save. |
| FR27 | **Session Handler** |
|  | SessionManager will create a Session after a client send a create request.  Right after the session is created, the session will be attached a TransmitHandler. The Session can recognize a TrackManager by the client FBID.  After the tempo setting is changed by the creator, the Session will notify all attendants to change their tempo.  The Session will sent the attendant list to the client if asked.  Session Handler will be attached a TransmitHandler**,** it will tell Session Manager to send to other members in the session. |
| FR22 | **Database** |
|  | The server will be connected to the database via a DBConnector class, which contains a set of java APIs that the server can call to manipulate data in various field within the database. The APIs consists of four methods for each database field: <1> create(), <2> delete(), <3> update/set(), and <4> find/get(). |

* + **Internal Interface Requirement**

**Client**

|  |  |
| --- | --- |
|  |  |
| IIR2 | Login Handler Module receives the information from Login Module to handle. |
| IIR3 | If Login Handler Module handles the information successfully, then it will get user’s id, user’s name and send packet to inform server though service handler. |
| IIR4 | If user login successfully, then it can display the main menu. |
| IIR5 | Invite Friends send request Friend List to Invitation Handler and get Friend List from Invitation Handler. After user choosing, Invite Friends will send the invitee List to Invitation Handler |
| IIR6 |  |
| IIR7 | After User accept one Invitation, Invitation will send the session number to Create Session to start Session |
| IIR8 | Create Session can create a Session Handler,  Session Handler will create Service Handler and register itself to the Service Handler.  Create Session will make Session Handler to send a request with creator FB ID and session name to Service Handler. Manager will add this <ID, Session Manager> to the Map. |
| IIR9 | When the tempo setting is changed by the creator, the session handler will tell Tempo to change |
| IIR10 | Invitation Handler can get the list of members and user being invited from session manager |
| IIR11 | Session Handler can ask Service Handler to create a Session for the user.  Session Handler can send a sound packet to Service Handler.  Whenever there is a sound by other member in the session, Service Handler will send a sound packet to Session Handler.  Whenever the tempo in the session is changed, Service Handler will tell Session Handler. |
| IIR12 | Notify Instrument List to show |
| IIR13 | Notify Instrument List to show |
| IIR14 | Play keyboard send sound to Instrument Player |
| IIR15 | Play keyboard can send a sound info to Session Handler. |
| IIR16 | Play drum pad send sound to Instrument Player |
| IIR17 | Play drum pad can send a sound info to Session Handler. |
| IIR18 | Send tempo increase/decrease to Session Handler |
| IIR19 | Whenever the tempo is changed by creator, tempo will be reset after certain time |
| IIR20 | Session Handler can send a sound info to Instrument Player to play a sound. |
| IIR30 | Service handler can send user information to Login Handler. |
| IIR34 | Login Handler can send user information to database to save. |
| IIR44 | Invitation can request Invitation List from Invitation Handler and get Invitation List from Invitation Handler. |

**Server**

|  |  |
| --- | --- |
|  |  |
| IIR27 | Reactor will monitor all the Acceptor registered to it. |
| IIR28 | After Acceptor initiate the ServiceHandler, Reactor will continuously monitor the ServiceHandler. |
| IIR29 | When the ISCONNECTABLE event is occurred, Reactor will call the handle method in the Acceptor to initiate a ServiceHandler and register ServiceHandler to itself. |
| IIR30 | LoginHandler will use ServiceHandler to communicate to Client. |
| IIR31 | InvitationHandler will use ServiceHandler to communicate to Client. |
| IIR32 | Service Handler can tell the Session Handler a user is leaving.  Service Handler can tell the Session Handler a user is joining.  Service Handler can send a music packet to Session Handler.  Service Handler can send a message packet to Session Handler. |
| IIR33 | RecordHandler will use ServiceHandler to communicate to Client. |
| IIR34 | Login Handler Module can store user information to database. |
| IIR35 | Invitation Handler can send user's request to Invitation Manager for retrieving his/her own invited list.  Invitation Handler can also call Invitation Manager to add participants to a session. After Invitation Manager creation, Manager will return ACK.  Invitation Handler can send user's response to specific Invitation to Invitation Manager.  Invitation Manager will remove them from the user's invited list and return ACK. |
| IIR36 | Session Handler can call Manager to create a Session Manager. Then Session Handler will register itself to the Session Manager. Session Handler will return ACK. Service Handler send request with Session ID and FB ID to Manager. The Manager will find the Session Manager by Session ID, and make the Session Manager create a Track Manager for the FB. Session Manager will tell database to add the FB id in the session row. |
| IIR37 | Once the session is finished, SessionManager will merge all the records of Track Managers into a record file and store it into database. |
| IIR40 | Invitation Manager can look up the Database to find if one has logined or not. |
| IIR42 | Session Manger will tell Database to create a row with Session name and Creator FB ID. Session Manager knows the row Index. |

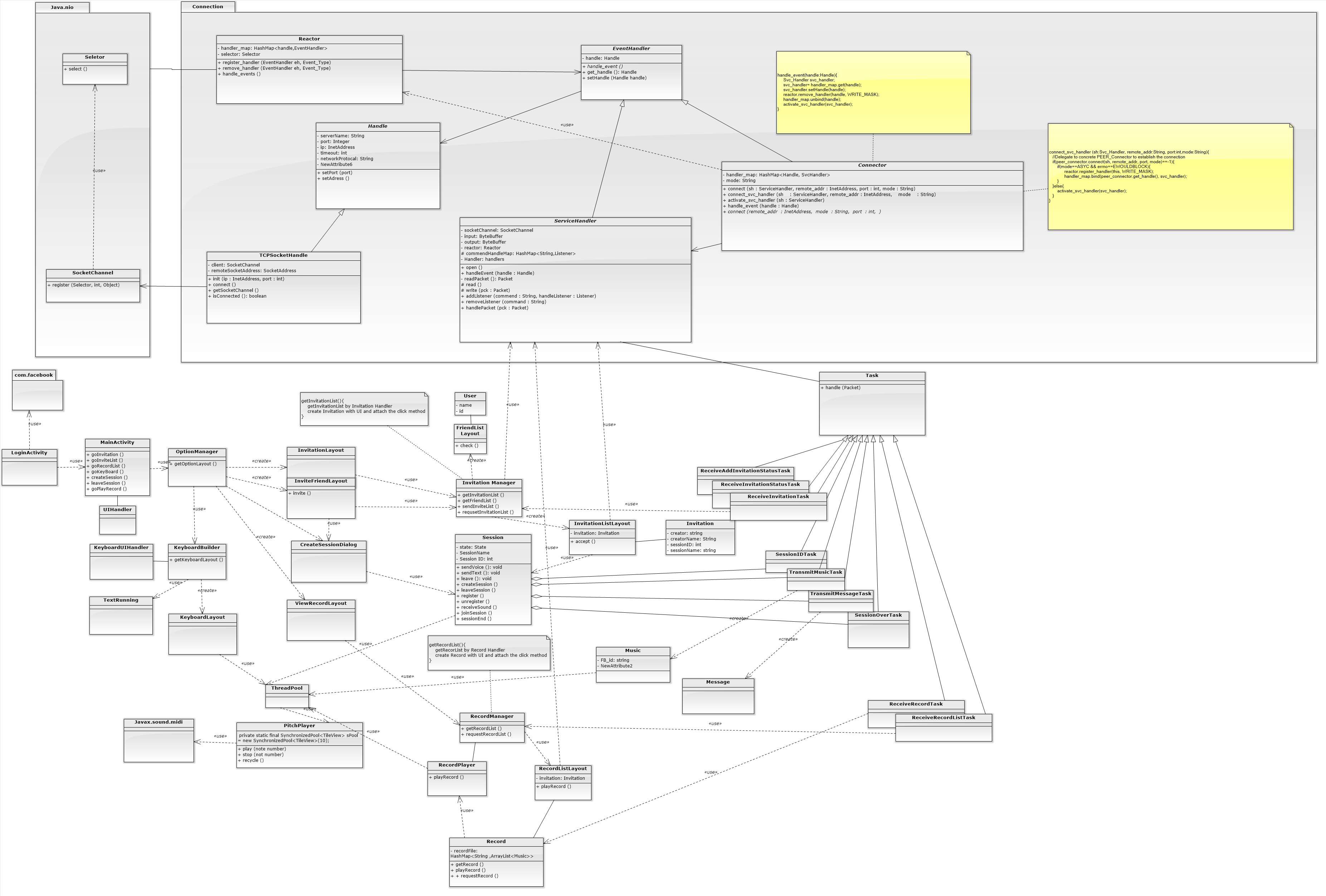
**External Interface Requirement**

**Client**

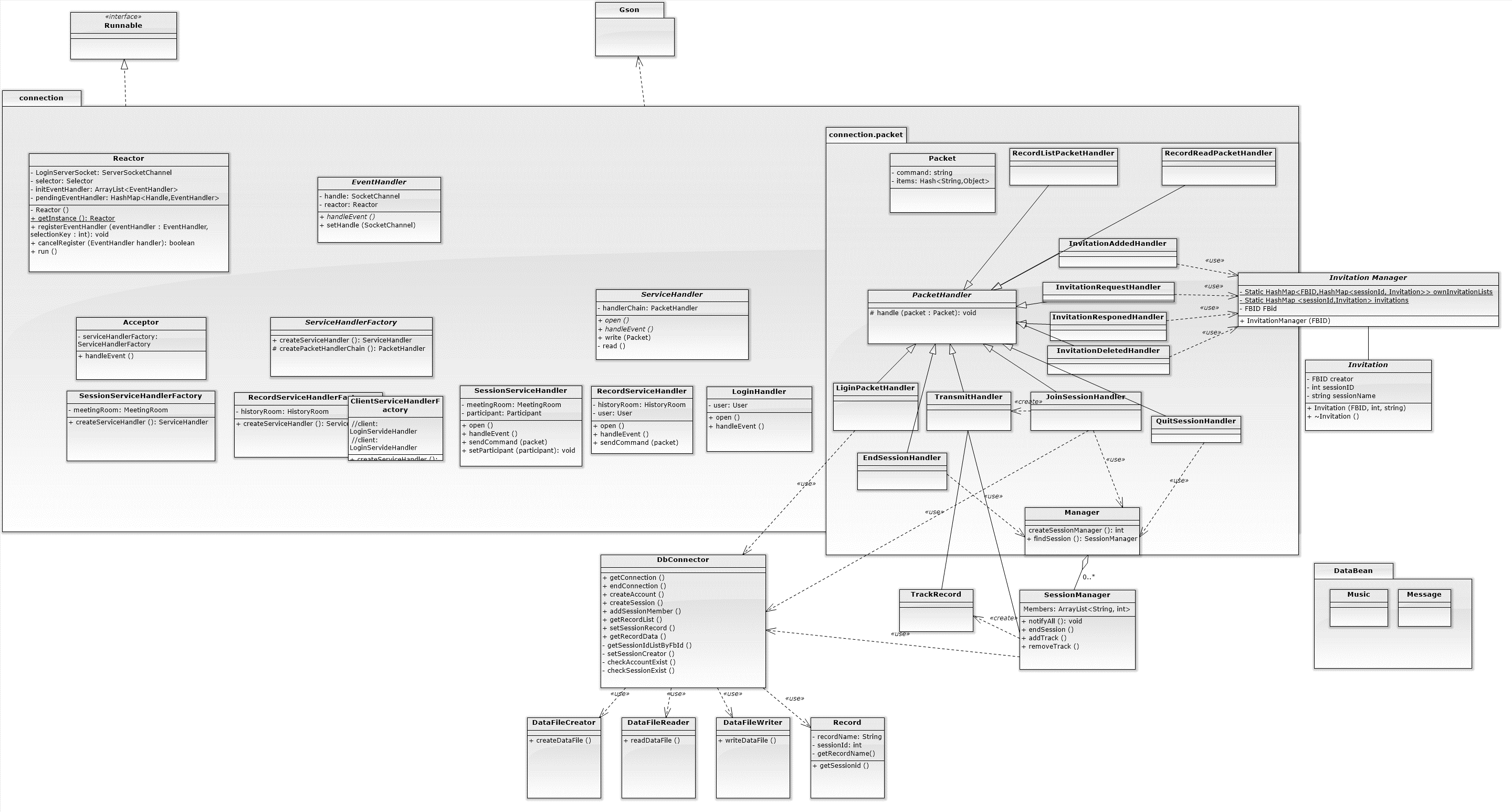
|  |  |
| --- | --- |
|  |  |
| EIR1 | User to UI |
|  | The interface between user and android application can touch application to start though Login Module. |
| EIR2 | Login to Facebook Server |
|  | Login Module sends Facebook Server on request (access token) and receives the information of acknowledgment from Facebook. |

**Class Diagram**

**Client:**



**Server:**



**Protocal**

**Client:**

//FB login (Acces\_token: get\_friend\_list)

/\*第一次跟Server 建立連線 以Login 為例

ServiceHandler loginHandler = AndroidApplication.getInstance().getLoginHandler();

Connector.connect(loginHandler, LOGIN\_PORT\_NUMBER);

\*/

)

send:Login (User->FB\_id,FB\_name,FB\_picture)

receive:Login\_Response Stats

//create user

//go to Lobby(default sate:single\_state)

-------------------------------

按下CreateSession Button後

send:Join\_Session FB\_id,FB\_name,Session\_Name

receive:Join\_Session Stats

//第一次跟Server 建立連線

//initiate session

//change Lobby state(Host\_state->CreateSession Button會變成Invite Button->傳送訊息Button(visible)，結束Button(visible))

按下Invite Button後

//update FB\_FriendLists

//choose friends

send:Add\_Invitation {FB\_id,Session\_id,[Friends\_id],Session\_Name}

recevie:Add\_Invitation\_Response Status

----------------------------------

按下Invitation Button 後

send:Request\_Invitation FB\_id

receive:Request\_Invitation\_Response Invitation\_List->{[sessionID, sessionName, creatorID, creatorName])

選定Invitation 後

send:Join\_Session Session\_id, FB\_id (InvitationHandler/SessionHandler)

receive:Join\_Session\_Response Status "OK"

//initiate Session

//change Lobby state(Acceptor\_state->CreateSession Button會變成"離開Button"->傳送訊息Button(visible))

------------------------------------------

按下Record Button

send:Request\_Record\_List FB\_id

receive: Request\_RecordList\_Response RecordList

//go to Record Activity

------------------------------------

Session/Invitation

Session Over

Invitation

send:Delete\_Outdated\_Invitation Session\_id

receive: Delete\_Outdated\_Invitation\_Response Status

Session

send:Session\_Over Session\_id

receive:Session\_Over\_Response Status

// disconnect the sessionHandler

Session quit

send:Session\_Quit Session\_id

//quit the session

receive:Session\_Quit\_Response Status

---------------------------------

Session

//SEND SOUND

send:Transmit\_music Music

receive:Transmit\_music\_Response Status

send:Transmit\_message Message

receive:Transmit\_message\_Response Status

//RECEIVE

receive:Transmit\_music\_Receive Music

No send:

receive:Transmit\_message\_Receive Message

No send:

receive:Session\_id\_Response Session\_id

No send

**Server:**

Login

receive:Login (User->FB\_id,FB\_name,FB\_picture)

send:Login\_Response Stats

//Add to DataBase

-------------------------------

Session

receive:Join\_Session FB\_id,FB\_name Session\_Name Session\_id

//if it is the creation, initiate session Manager

//if it is join, add the SessionHandler to corresponding session manager

send:Join\_Session\_Response Stats

receive:Session\_Over Session\_id

//send all participant he will be kick

send:Session\_Over\_Response Status

receive:Session\_id\_Response Session\_id

No send:

receive:Session\_Quit Session\_id FB\_id

//remove the SessionHandler from Session Manager

No send:

receive:Transmit\_music Music

send:Transmit\_music\_Receive Music //SEND TO OTHERS EXCEPT THE SENDER

receive:Transmit\_message Message //SEND TO OTHERS EXCEPT THE SENDER

send:Transmit\_message\_Receive Message

send:Session\_id\_Response Session\_id

No receive

----------------------------------

Invitation

receive:Request\_Invitation FB\_id

//go to database to find corresponding invitation\_List

send:Request\_Invitation\_Response Invitation\_List->{[sessionID, sessionName, creatorID, creatorName])

receive:Delete\_Outdated\_Invitation (FB\_id Session\_id)

//go to database to delete corresponding Invitation

send:Delete\_Outdated\_Invitation\_Response Status

receive:Join\_Session (Session\_id, FB\_id)

//go to database to delete corresponding Invitation

send:Join\_Session\_Response Stats

--------------------------------------------------------

Record

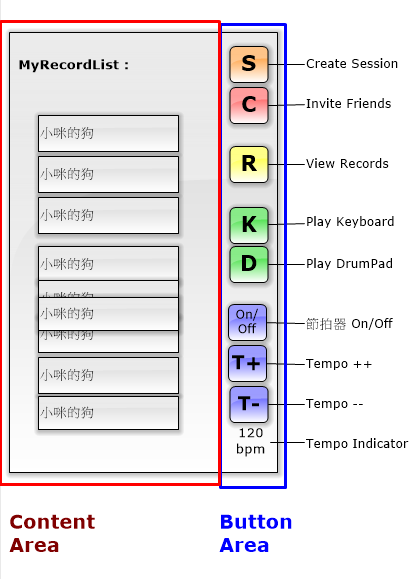
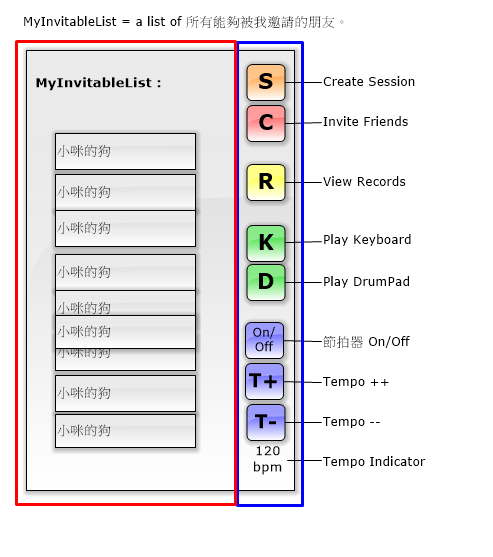
receive:Request\_Record FB\_id

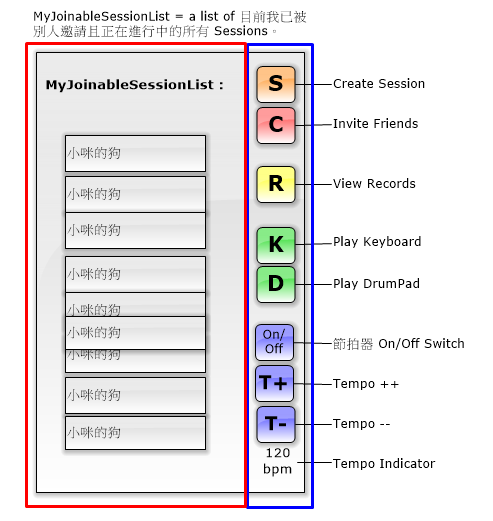
// Retrive Record from database

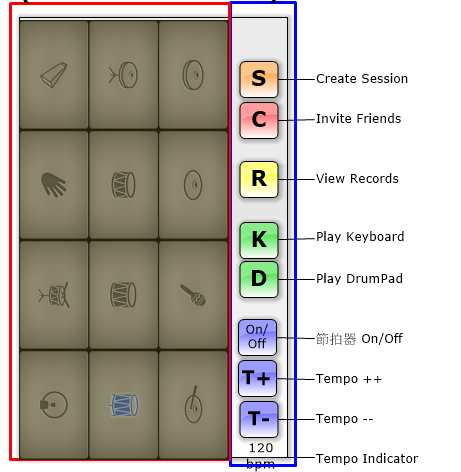
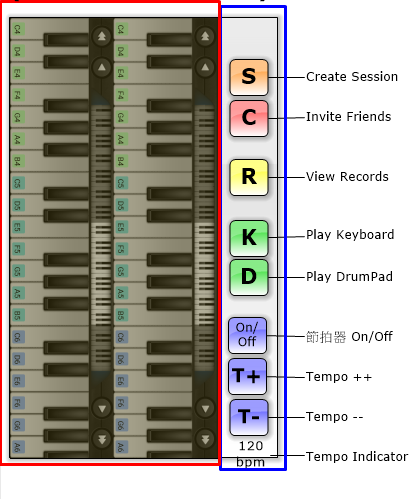
send:Request\_Record\_Response Record

------------------------------------

**UI Design**

** **

****

****