Developers of the Millenium

System Design

Table of Contents
CRC Cards
System Architecture Design

MainActivity	FragmentChangeListener
• Starting Login page • Title page	• FragmentChangeListener

LoginPageFragment	Fragment
• displays the login page and would later send the information to the server	• RegistrationPageFragment

RegistrationPageFragment	Fragment
• This page allows the user to either login, or register to the site. A client/user will also have the option to proceed as a guest but with limited functionality.	• LoginPageFragment

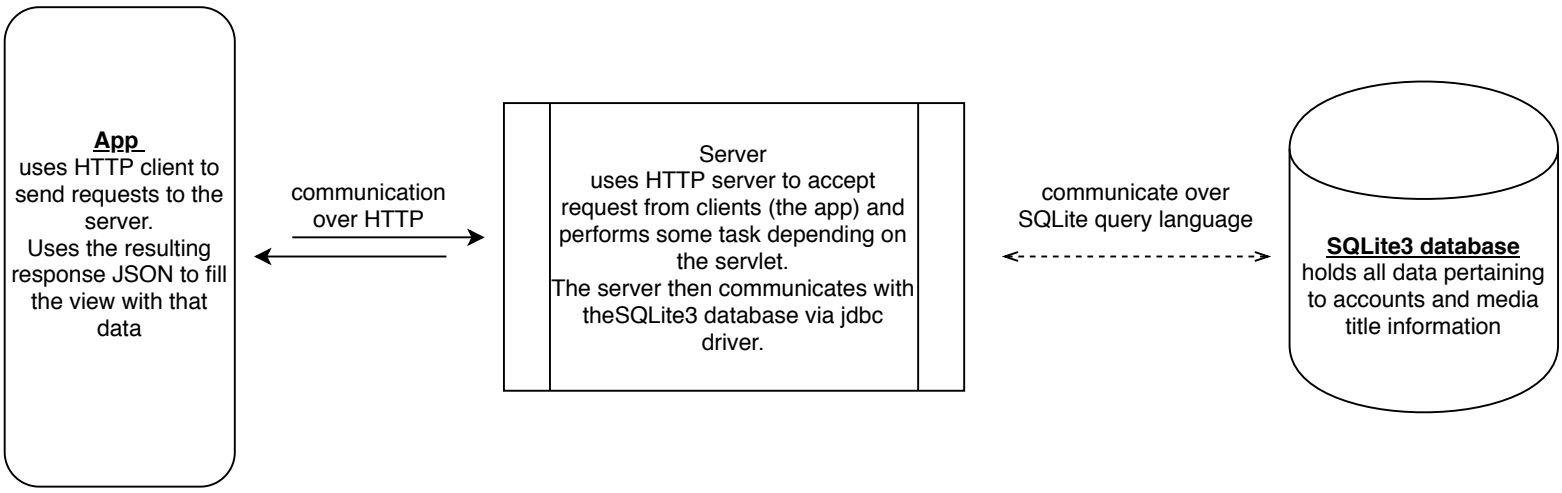
MediaProfilePageFragment	Fragment
• Shows the overall profile of the selected media. This includes current rating, media art , Title and the ability to add to favorites.	• ReviewsFragment • MediaProfileFetcher

ReviewsFragment	Fragment
• This allows a user/client to give a review on the selected media.	• MediaProfilePageFragment

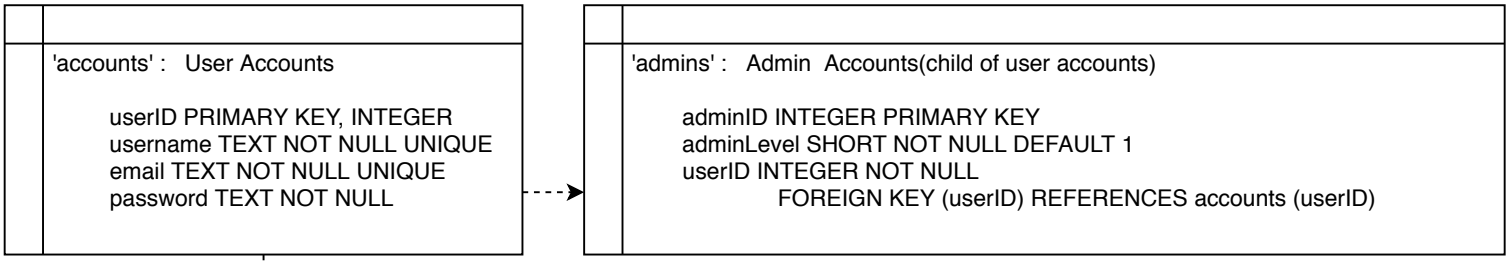
FragmentChangeListener	
• Watches for changes in the fragment backstack	• MainActivity

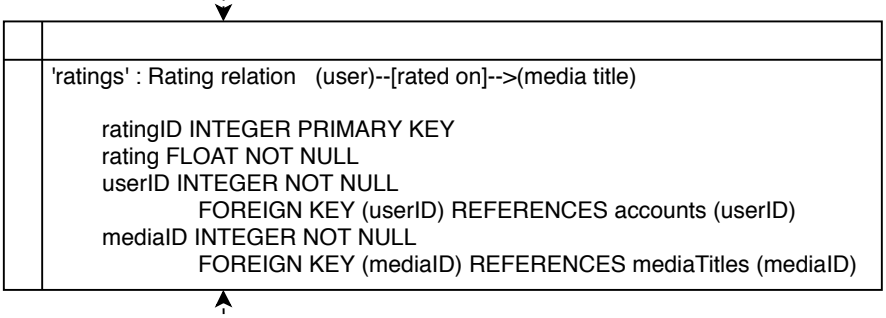
MediaProfileFetcher	
• Fetches information about the Media Profile from database on a server	• MediaProfilePageFragment

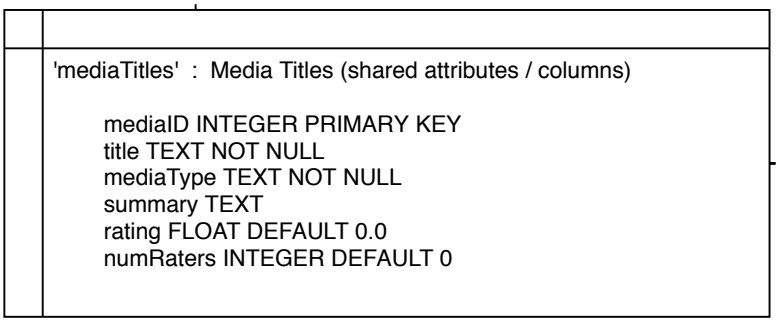
Basics



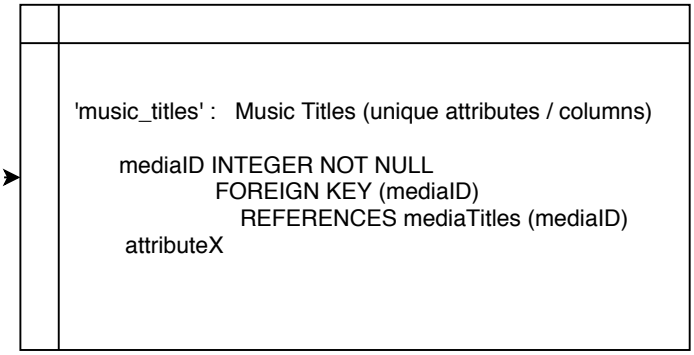
Database (this is a work in progress)

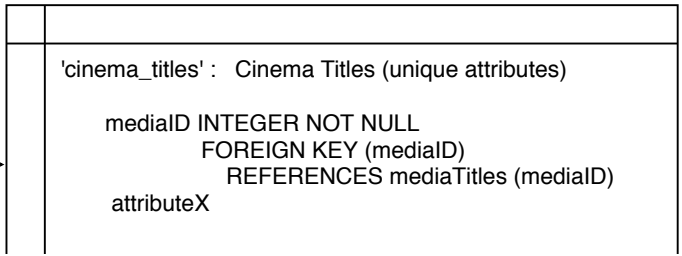


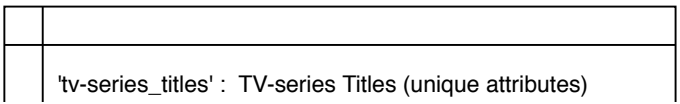




use media types {'cinema', 'music', 'tv-series', 'video game', 'novel'}







mediaID INTEGER NOT NULL
FOREIGN KEY (mediaID)
REFERENCES mediaTitles (mediaID)
attributeX

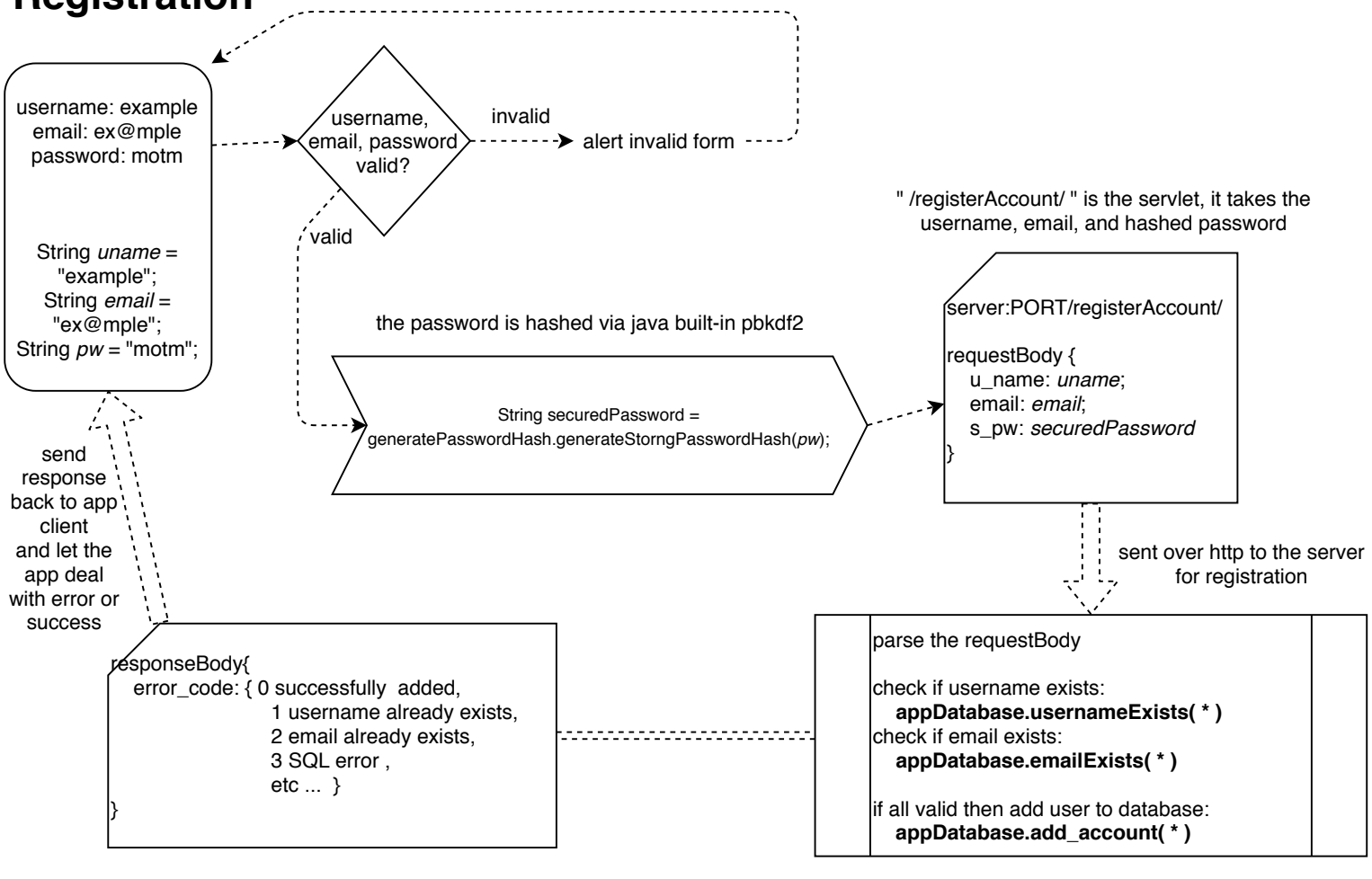
'videogame_titles' : Video Game Titles (unique attributes)

mediaID INTEGER NOT NULL
FOREIGN KEY (mediaID)
REFERENCES mediaTitles (mediaID)
attributeX

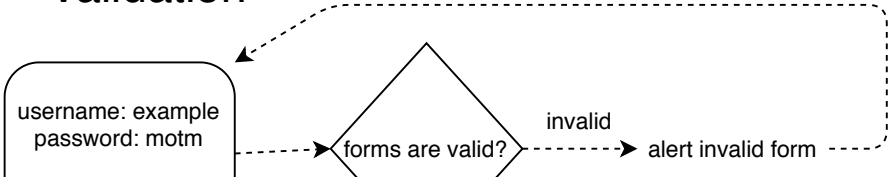
'novel_titles' : Novel Titles (unique attributes / columns)

mediaID INTEGER NOT NULL
FOREIGN KEY (mediaID)
REFERENCES mediaTitles (mediaID)
attributeX

Registration



Validation



or

email: ex@mple
password: motm

String *uname*;
String *email*;
String *pw*;

valid

the given password is hashed via java built-in pbkdf2

String *securedPassword* =
generatePasswordHash.generateStorngPasswordHash(*pw*);

alternatively use the wrapper function that achieves the same thing

" /loginAccount_u/ " is a servlet, it takes the username and hashed password.

alternatively

" /loginAccount_e/ " is a servlet, it takes the email and hashed password.

server:PORT/loginAccount_e/

requestBody {
email: *email*;
s_pw: *securedPassword*
}

server:PORT/loginAccount_u/

requestBody {
u_name: *uname*;
s_pw: *securedPassword*
}

sent over http to the server for validation

parse the requestBody

check if username or email exists:
appDatabase.usernameExists(*)
appDatabase.emailExists(*)

get the userID that that matches username or email:
appDatabase.get_user_from_name(*)
appDatabase.get_user_from_email(*)

using userID, check if the password matches:
appDatabase.validatePassword(*)

if valid password then create a session token to send:
function not defined

send response back to app client and let the app deal with error or success

responseBody{
error_code: { 0 verified,
1 username doesn't exist,
2 email doesn't exist,
3 invalid password,
4 SQL error ,
etc ... }
session_token: ~~~~~
(?) userID : ~~
}

get collection of Media titles (for example top 10)

app
media collection
page

depending on the type of data we need, we send a request through a specific servlet, one which probably wont take a requestBody unless there is a specific criteria to meet (for example get titles of a specific genre or titles with a certain tag keyword

server:PORT/get???/

send the request over http

parse the requestBody (if any, depending on servlet)

get the collection of media titles via some server function which will query the database and return a collection:
appDatabase.get???(*)

send response back to app

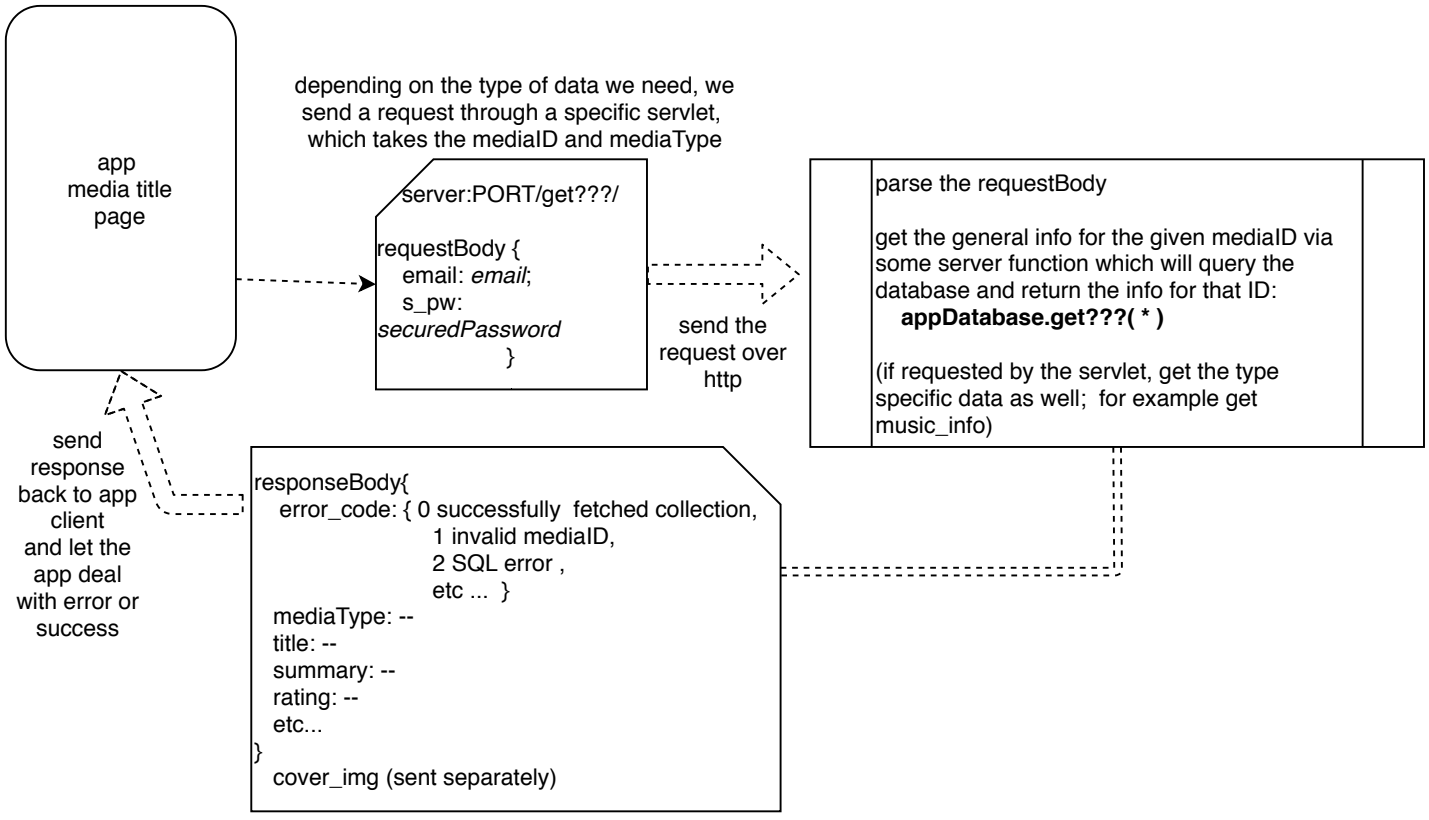
responseBody{
error_code: { 0 successfully fetched collection,
1 didn't find any collection

client
and let the
app deal
with error or
success

2 SQL error ,
etc ... }
list_of_titles: [title1_id, title2_id, ... , titleN_ID]
or
list_of_title_JSONs: {
{id: title1_id, mediaType: --, etc},
.... ,
{id: titleN_id, mediaType: --, etc}
}
(optionally) mediaType: sometype
}

send only the title IDs (the other data will
be fetched separately given the IDs)
or send a list of json which contains the
ID and relevant data

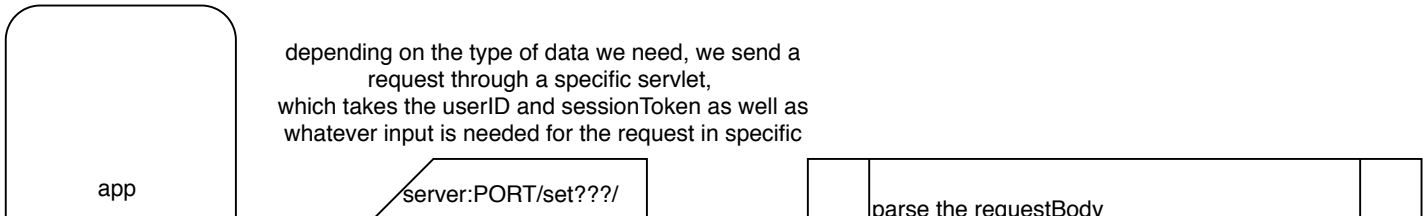
get Media title info

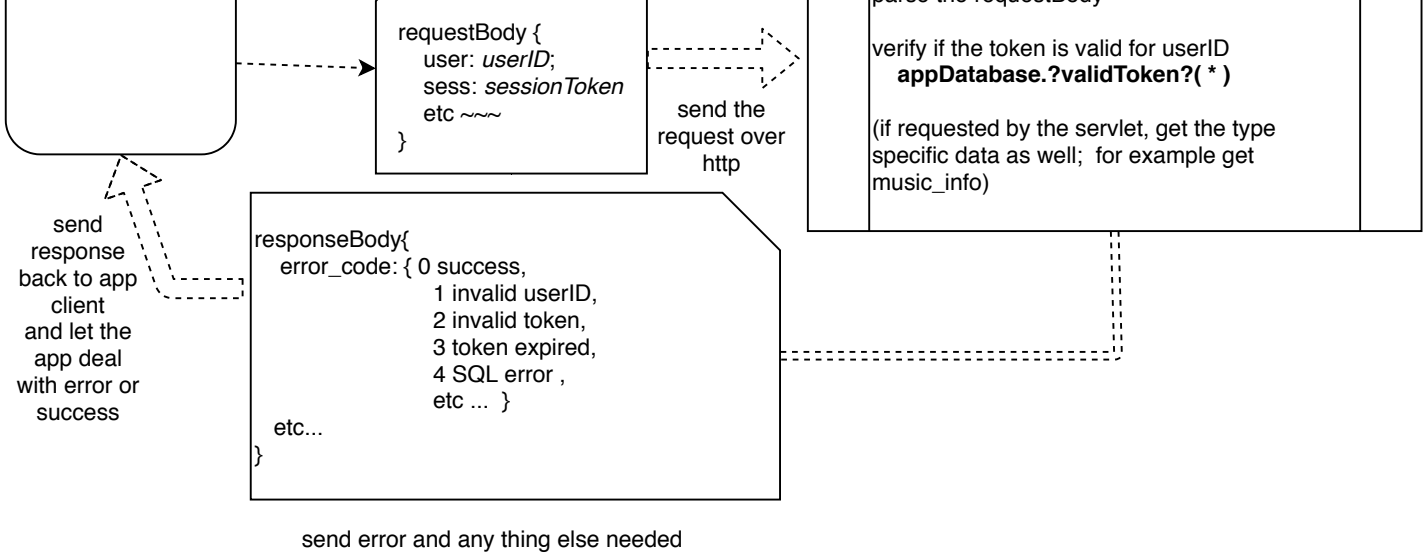


send all necessary data (image will
probably have to be sent separately due
to http functionality)

make account sensitive request (WIP)

(post review, set rating, make changes to account page, etc)




requestBody {
user: *userID*;
sess: *sessionToken*
etc ~~~
}
send the request over http
verify if the token is valid for userID
appDatabase.?validToken?(*)
(if requested by the servlet, get the type specific data as well; for example get music_info)
send response back to app client and let the app deal with error or success
responseBody{
error_code: { 0 success,
1 invalid userID,
2 invalid token,
3 token expired,
4 SQL error ,
etc ... }
etc...
}
send error and any thing else needed