

PlayTime

1. Introduction

Sports are a crucial part of an individual's growth, fitness and development. Inspired by playing many sports ourselves, we came across the problem of finding sport players for non-single sports to partner up with. Therefore, PlayTime is an app that aims to help users find players or certified coaches nearby. With the help of this app, you can find people at your skill level. You can view profiles, user galleries and match with potential players or coaches and plan sessions with them over private chat. This report aims to provide a detailed description on the requirements, design, implementation and evaluation of PlayTime.

2. Design and Implementation

2.1 Requirements Analysis

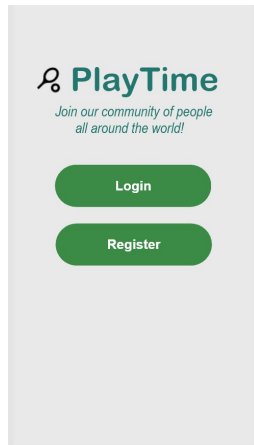
After discussing on the features of the app in detail, we have decided to pursue the following requirements:

1. Be able to create an account by registering with an email address.
2. Be able to login to an existing account using email and password with prebuilt data validation.
3. Be able to edit and change information listed in the user profile such as profile image, name, phone number and sports played.
4. Be able to view potential matches in the home page and swipe right to accept the user being displayed and swipe left to decline. The user's profile image along with information on the sports they play will be displayed on a card.
5. When matched with a user, i.e, two users swipe right on each other, be able to connect with them through private chat.
6. Private chat is encrypted before being stored in the database for privacy concerns.
7. Create a gallery for the user to upload images from their phone
8. View potential match's gallery before swiping.

2.2 Design

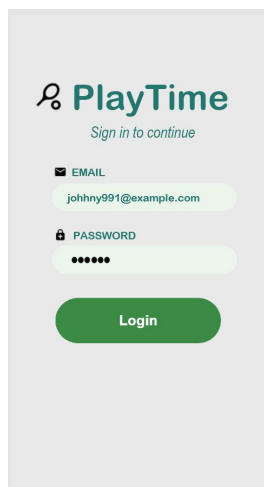
The activities have been designed with AdobeXD and then implemented with XML on the android layout. We have attached the Adobe XD file of the design for your reference in the main folder as well as screenshots of the activities below:

Activity Design:



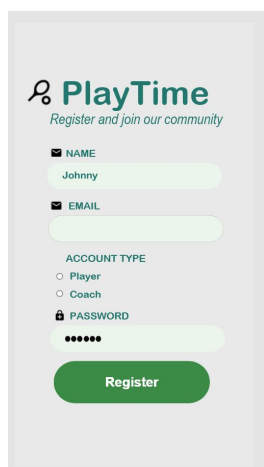
ChooseLoginOrRegistration Activity : Startup page where the user either logs in with an existing account or chooses to create a new account.

We used Material design icons and designed the buttons using XML. We also imported some of the fonts from FontAwesome to make the design outlook striking.

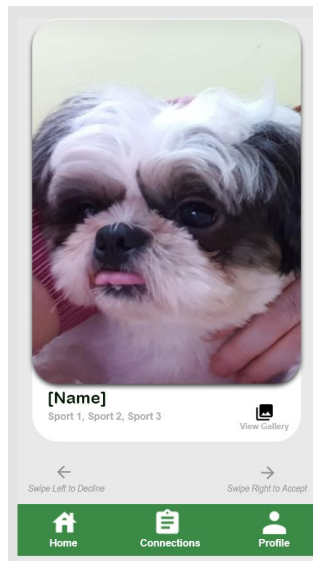


LoginActivity: User enters login credentials and gets authenticated.

The user needs to add their email in the email Edit Textbox and their password in the password Edit Textbox. For data validation we used the “Input Type” attribute of XML for each textbox making the input types email and text password accordingly. Furthermore, for data verification we used firebase authentication.



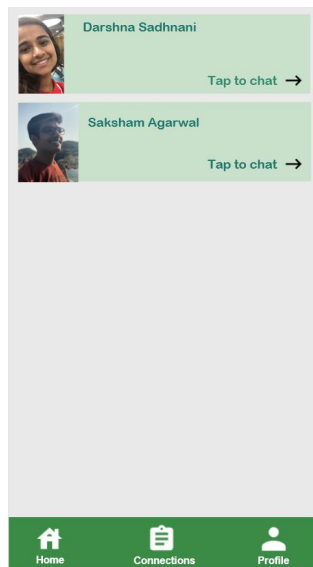
RegistrationActivity: User registers an account with play.time. This is where the user chooses to be a player or a coach. The user will be able to input his name, email, account type as well as password. Data validation will be automatically done by firebase authentication libraries as well as input type from android studio. Preceding this activity, users will be redirected to their profile page in order to complete filling in their personal details.



MainActivity: The main activity shows potential matches where the user can swipe left or right. a settings tab also present which would create an intent to access the app_settingActivity. App_settingActivity will allow the users to store their custom settings in shared preferences.

Cards: This is the object which contains the information of the potential match and is used to populate the arrayAdapter.

ArrayAdapter: Consists of a list of cards which will be displayed on the main activity homepage. When a user registers, they will be added on this list.

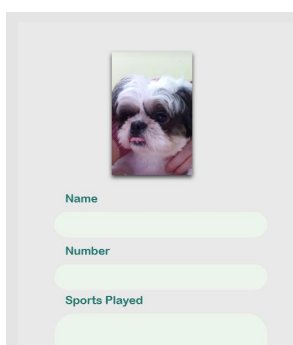


MatchesActivity: The matches activity contains a list of all the matches of the user. When two users mutually swipe right on each other, they appear in each others' MatchesActivity section. Thereon, the user can click on the match to view its information, and can start chatting with the match (intent ChatActivity.class).

MatchesObject: This is used to obtain the information of the match and is used to populate the MatchesAdapter.

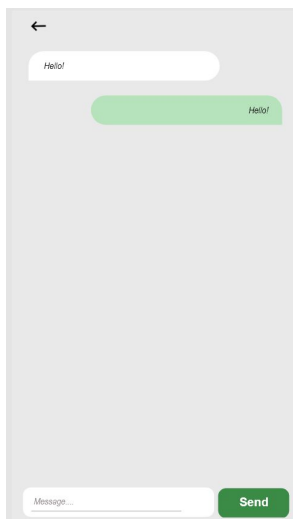
MatchesAdapter: This is used to populate the RecyclerView with the data from MatchesObject.

MatchesViewHolders: object that represents each item.



SettingsActivity: The SettingsActivity serves two functions - the user can view their own details, and they can change those details

as well. It fetches the user information from Firebase and displays it for the user to view. And it relays information back to Firebase in case the user has decided to change or add any information (e.g. profile picture).



ChatActivity: Matches can talk to each other using the chat interface. The current user and match have different coloured chat boxes and text for differentiation. As the user sends the message, there are background processes running to store the message in the database, and to display the message in the chat box.

The ChatActivity uses several helper classes as well.

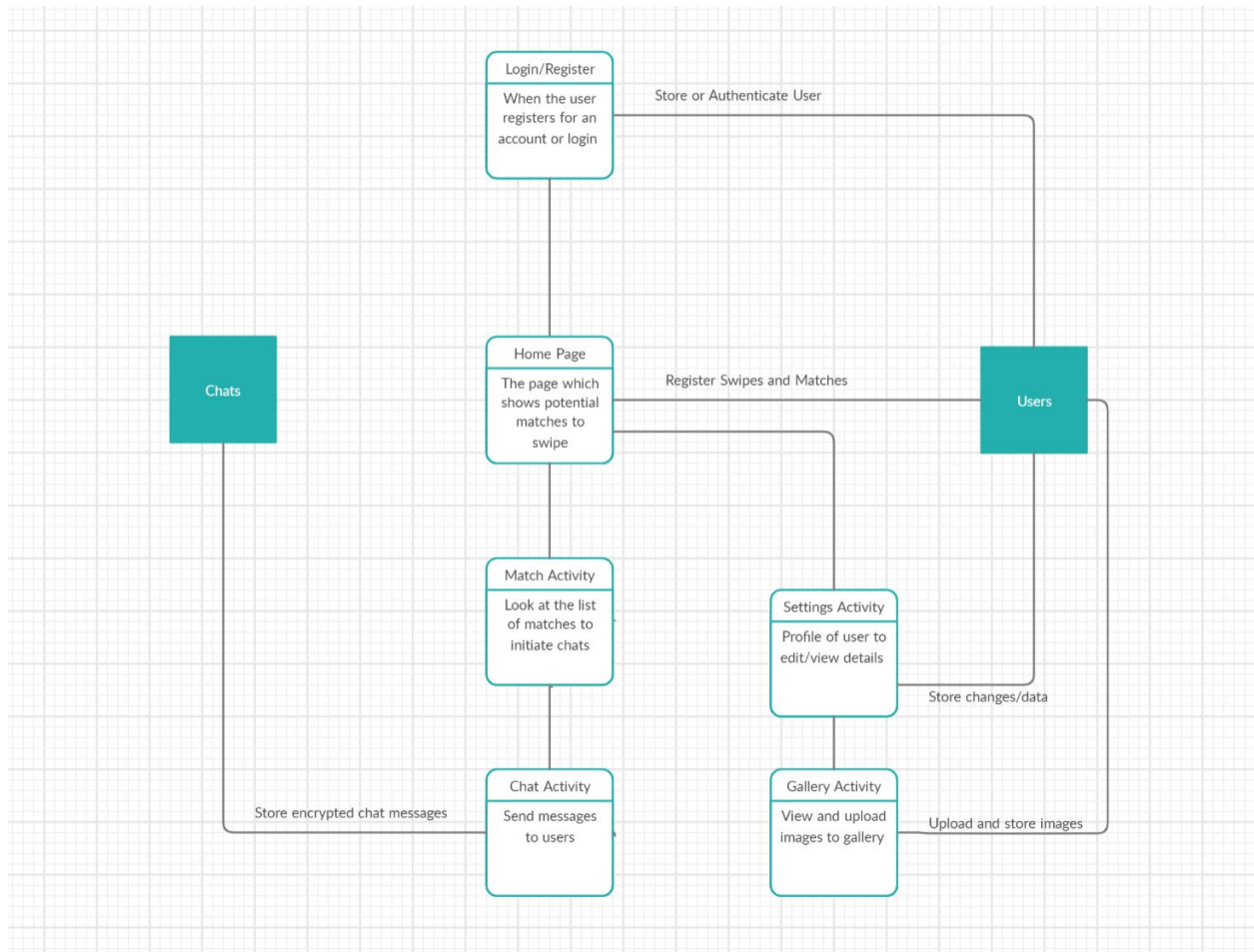
Chat object: This is used to obtain the chat information and is used to populate the ChatAdapter.

ChatAdapter: This is used to populate the RecyclerView with the data from MatchesObject.

ChatViewHolder: It is the very object that represents each item in our collection and will be used to display it.

A background service will be running in this activity which would indicate any changes happening (events addition or removal) in the firebase database.

Interrelation of Activities with each other and database illustration:



User Interface:

We are planning to use the Material design user palettes, layout templates and components to improve user experience of the app.

Add notifications for the app when the user gets the matches using notification channels.

External open source libraries:

Diolor - SwipeCards

2.3 Implementation

- **Storage of Data in the Application**

Since we will be providing users a functionality of setting up accounts with our application, we will need a database service which will be used to store user's information such as login credentials. We decided on incorporating Firebase realtime database with our application, it allows developers to build rich, collaborative applications by allowing secure access to the database directly from client-side code. It can also be used for authentication purposes for user logins. We will also use firebase to store our Event specific data.

Login/Register: We use FirebaseAuth to Authenticate users. We use FirebaseUser to create a user and store in the database upon registration. The user inputs information like name, phone number, sports, and can even upload a profile picture which is stored in the database.

Register the Swipes to the database : Create a connection between the users when the user swipes left or right. If the player swipes left on the coach then the coach's user id will be added to the "Denied" child of the current user. If the player swipes right on the coach then the coach's user id will be added to the "Approved" child of the current user.

Register the Matches to the database: Each user will have a reference to the people they swipe right on. If the person they swipe right also swipes right on them, they get added to the list of the user's matches, and a chat id is created for the two of them.

Chat: Create a child called chat. This child will have a separate child for every interaction that the user will have. So, each chat reference corresponds to an interaction that two users are having. Therefore, a pair of matches will have the same chat id.

The chat id will store a reference to every message sent in that particular interaction between two matches, and will store details such as the content of the message, and the userid of the user that sent the message.

3. Testing and Evaluation

We used two types of user testing on our application: Espresso and Acceptance testing.

LoginActivityTest: This espresso test, does UI testing for the login activity page and what happens after that. First, the email and password is entered, then login is pressed. After that, the bottom navigation bar is tested by clicking on “Connections” and “Profile”.

RegisterActivityTest: This espresso test does UI testing for the Register Activity page and what occurs after that. First the registration details are put into the fields and the register button is clicked. Then, we are able to view the profile activity and fill in the fields over there. We save the changes to the profile. Then after that, we go to the connections page and the home page. Finally the signout button is clicked.

ChatActivityTest: This espresso test does UI testing for the Chat Activity page. First, the user account is logged into. Then we go to “Connections” to see the user’s pre-existing matches. We go to one of the matches and type in a message to see how it is sent. Then the back button is pressed which takes us back to the connection page. Furthermore, we view the profile, the home page and sign out before the test ends.

Test Summary

4	0	2m34.53s
tests	failures	duration









100%
successful

Packages

Classes

Class	Tests	Failures	Duration	Success rate
com.example.playtime.ChatActivityTest	1	0	1m6.26s	100%
com.example.playtime.ExampleInstrumentedTest	1	0	0.026s	100%
com.example.playtime.LoginActivityTest	1	0	47.782s	100%
com.example.playtime.RegisterActivityTest	1	0	40.460s	100%

debugAndroidTest

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.example.playtime		79%		64%	58	145	86	342	26	92	3	28
com.example.playtime.Chat		93%		50%	17	37	16	106	8	28	0	7
com.example.playtime.Matches		95%		62%	11	35	10	93	5	26	0	7
com.example.playtime.Cards		91%		70%	6	16	6	39	3	11	0	2
Total	439 of 2,828	84%	54 of 144	62%	92	233	118	580	42	157	3	44

Analysis of Test coverage: The reason why the test coverage is around 60-80 percent is because espresso testing does not include the swipe card library since it is not

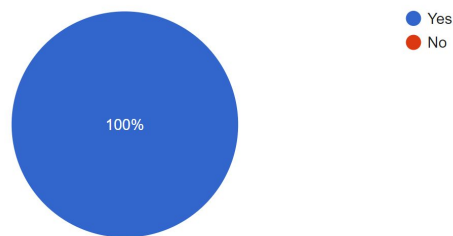
considered a “view” in the layout. However, in order to test this, we have also included some user acceptance tests and asked the users to fill a survey. We got the following survey results shown below.

User acceptance test statistics:

We asked at least 10 users to fill out the user acceptance test and asked the following questions, and these were the response statistics. We added very generalized yes/no questions along with one long answer to ask for what users would like to see in the next update of the app. We added the survey link in the references.

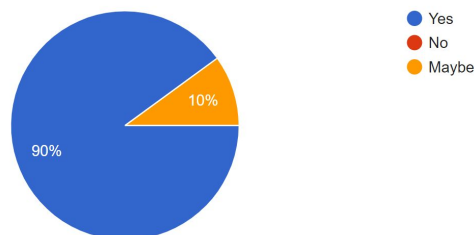
Did you like the idea of the app?

10 responses



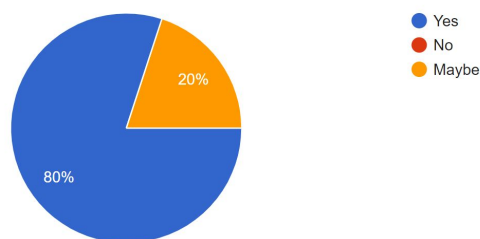
Did you find the User Interface design good?

10 responses



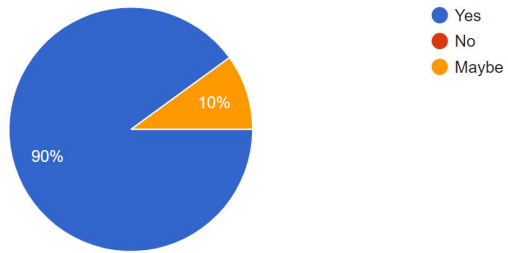
Did you find the User Interface smooth and user friendly?

10 responses



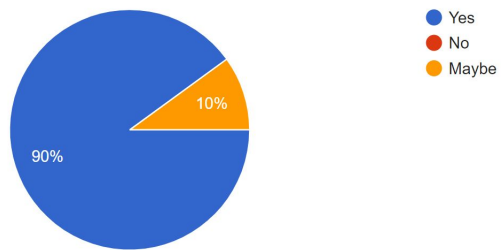
Did you find the chatting feature secure and smooth to use?

10 responses



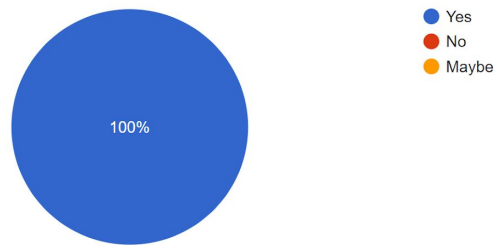
Did you like the profile viewing option?

10 responses



Did you like the idea of swiping to find a player/coach?

10 responses



Are there any changes you would like to make to the app in the next update?

7 responses

More people

It is perfect

Yes make it faster

no

Maybe make the UI design more creative

It could allow us to schedule facility booking through the app too

The criteria to judge a players capability to play can be improved, since just videos might not accurately tell the actual playing talent of a person

Looking at the acceptance test results, most of the feedback was positive. Some feedback was received about the app being a bit slow, make the UI more creative, add a booking facility to the app and also improve the criteria to judge players/coaches.

Evaluation of the project requirements:

The following goals were achieved in the project from the requirement section:

Requirement	Achieved / Not Achieved
Be able to create an account by registering with an email address.	Achieved
Be able to login to an existing account using email and password with prebuilt data validation.	Achieved
Be able to edit and change information listed in the user profile such as profile image, name, phone number and sports played.	Achieved

Be able to view potential matches in the home page and swipe right to accept the user being displayed and swipe left to decline. The user's profile image along with information on the sports they play will be displayed on a card.	Achieved
When matched with a user, i.e, two users swipe right on each other, be able to connect with them through private chat.	Achieved
Private chat is encrypted before being stored in the database for privacy concerns.	Achieved
Create a gallery for the user to upload images from their phone and view matches gallery	Not Achieved

4. Conclusions

Result of the project:

We have achieved all of the requirements stated in section 2.1 of the report. We added a user login/registration, was able to match users and display potential users by swiping on their card: swipe right to accept and swipe left to decline. Furthermore, we were able to achieve making a private chatting system for matched users. We were also able to achieve displaying and making changes to the user profile such as changing their profile picture and other personal detail changes.

The shortcomings of the project are:

We could have implemented more features such as the gallery feature to view different player's pictures and videos. However, we realized that the realtime database that we used has a limit to image and video size storage. Furthermore, it was very slow when it came to retrieving. So, we could not successfully implement it in this update. Furthermore, instead of using only the firebase Realtime database we could have also

used the cloud firebase database since it is more efficient code-wise and may make the app potentially faster to use. We also would like to work on the reviews given by the user acceptance tests such as add a rating system to have a better judging capability for users to choose players, add a booking scheduling system and also make the app work faster and be less laggy, and make a more creative UI design. However, we are committed to make the following changes in the next update of our application.

5. References

Give references to any material / websites / books etc. relevant to your project.

SwipeCards Library: <https://github.com/Diolor/Swipecards>

Firebase: <https://firebase.google.com/docs/android/setup>

Material Design: <https://material.io/>

Youtube video tutorial :

<https://www.youtube.com/watch?v=MUiZhCUHXhk&list=PLxabZQCAe5fio9dm1Vd0peIY6HLfo5MCf&index=10>

User Acceptance Survey: <https://forms.gle/4u5JHS882btDKy5o7>

Contributors:

SADHNANI, Darshna Girish; darshna.sadhnani@gmail.com

AGARWAL, Saksham; 20462943; sagarwalad@connect.ust.hk