MobServ Challenge Project 2017

Ready2Meet: Event Organizer for Android

Berkay Köksal Alexander Küchler Saad Lamdouar

January 17, 2018

1 Application Description

The event organizer (name: Ready2Meet) is an Android app to facilitate creating events and inviting people. The core of the application, namely organizing an event, can be used in two different modes:

1. Planned event
2. Spontaneous event

For both modes, the user creates an event (including a start and end time, location, type of event and eventually a name). In the planned event mode, the user can invite specific people (friends) while for the spontaneous event, it is also possible to send an invitation to people which are close to the user's current location. Like this, he is able to spontaneously gather people based on their location.

Apart from this, all people joining an event should be able to share pictures of the event in the app and influence the event's outcome e.g. by voting for music on a party. Also, a chatroom should be available to communicate with the people before or after the event.

For all kinds of events, the organizing person can add a list of required material (e.g. drinks or food) so that the guests can register to bring some of the material and thus easing the organization.

Finally, every user invited to an event is provided additional information like the weather forecast for outdoor events and whether he is available on the date when the event is scheduled.

2 Features

The features of our event organization app include

- Create an event (e.g. party, sports, hiking, lunch, ...)
  - Invite friends
  - Invite people in a close area. I.e., the user can specify a radius to invite people and the people are notified if they are interested in this kind of events
- Chatroom for event
- Enable music organization/voting during the event
- Share pictures of the event
- Who brings what?
- Get some additional information for the event
  - Weather forecast for outdoor events
  - Check if timeslot is free in your calendar
- Share the event in other networks e.g. on facebook

We therefore plan to use Google API (maps), firebase database to provide a real time database, and OpenWeatherMap API to retrieve the weather forecast.

3 Developer Team

The team consists of three people:

- Saad Lamdouar: UI developer
- Berkay Köksal: Core developer
- Alexander Küchler: Core developer

4 Business Model

Different business models are possible to push the app on the market. In this section, we discuss potential models and pricing strategies and finally chose the most promising model. Furthermore, we discuss a marketing strategy for our application.

4.1 Monetization

A completely free (e.g. open source) app is the first possibility. While this potentially targets the highest number of users as the app is free to use and accessible also in alternative stores like F-Droid, the application would not lead to any profit.

A second option is to use advertisements in the app in order to gain from the application. While this leads to a higher income, it contradicts the open-source strategy and thus would probably lead to a smaller target group. However, as the vast majority of Android users install apps from the Google Play Store, the decrease of users is acceptable.

Together with the app using advertisement, it is possible to offer an for a small price and without ads. An app which has only a priced version does not appear to be promising to us as most users will not be willing to pay for the service.

Also, In-App sales would be possible e.g. to enhance the possibilities of a user when creating events. In especial, this makes sense for business customers that could use the application to advertise their business. A possible scenario is creating events in the application which reflects the companies product portfolio (e.g. guided tours, sports, parties in a bar, ...). Like this, the company can acquire new customers which are nearby e.g. during their holidays and thus our app presents a new marketing strategy. In the scenario of commercial users, it would be possible to earn a reward for every user which joins the respective event due to our platform. The reward could be negotiated as a percentage of the tickets for the event or as a fixed amount. However, both cases would require to include a ticket-selling system for the billing. This would be a possible extension of the app.

In our case, the most promising strategy is to offer a free app with advertisements as well as a paid version without ads. In-App sales which target commercial customers

of the app can make the app more powerful. E.g. an event could be permanent or the radius to invite people could be extended. Also, the creators of an event could have additional possibilities.

4.2 Marketing Strategy

As our app aims at making events available to the highest possible number of potential participants, organizing events is the main possibility to advertise the app.

First, it is possible to offer free events to users which are organized by us. Second, we could offer discounts to our users for paid events which are organized by a third party. This is beneficial for both, the organizing party as they get additional participants and us, as we can make our app attractive for many users and thus gain a high number of users.

Both techniques aim at attracting as many users as possible. Even if we have some expenses for the events and eventually for offering the discounts, this can decline once we have sufficient customers.

4.3 Competitors

In this section, we shortly present a couple of competitors. Please note that the list is not complete and only presents a rough overview of the market.

Fever generates an event list taking into account the user's interests. However, it is only available in a few big cities and does not offer many events. Also, the focus is on commercial events while we aim at providing a platform to offer any kind of events, in especial, "private" and free ones.

WeTorch is a free app that provides a selection of events which are close to the user's location. It only exists in spanish and the focus lies on cultural events as well as arts.

Event Manager and Event Manager by Billeto give overviews of the event for event managers and help event organizers to grow their event reach and manage it.

While offering a platform is not the main focus, Facebook provides the functionality to invite friends to events or make them publicly available. In especial due to the large user community, it is an important competitor on the market.

5 UI Design and Click Stream

In this section, we present the click stream and thus the complete UI design of our app. Figure 1 shows the overall click stream with all possible links between views. The app is designed such that every page can be reached with only 2 clicks after starting the app.

The first screen of the app is the login page. Once a user has registered and logged in for the first time, the user credentials are stored so that this step is skipped in further uses of the app.

Once the user is logged in, the start screen of the app is designed to provide an overview of interesting events for the user. We display all events the user might be interested in according to its last search results in a map. Also, the user's current location is used

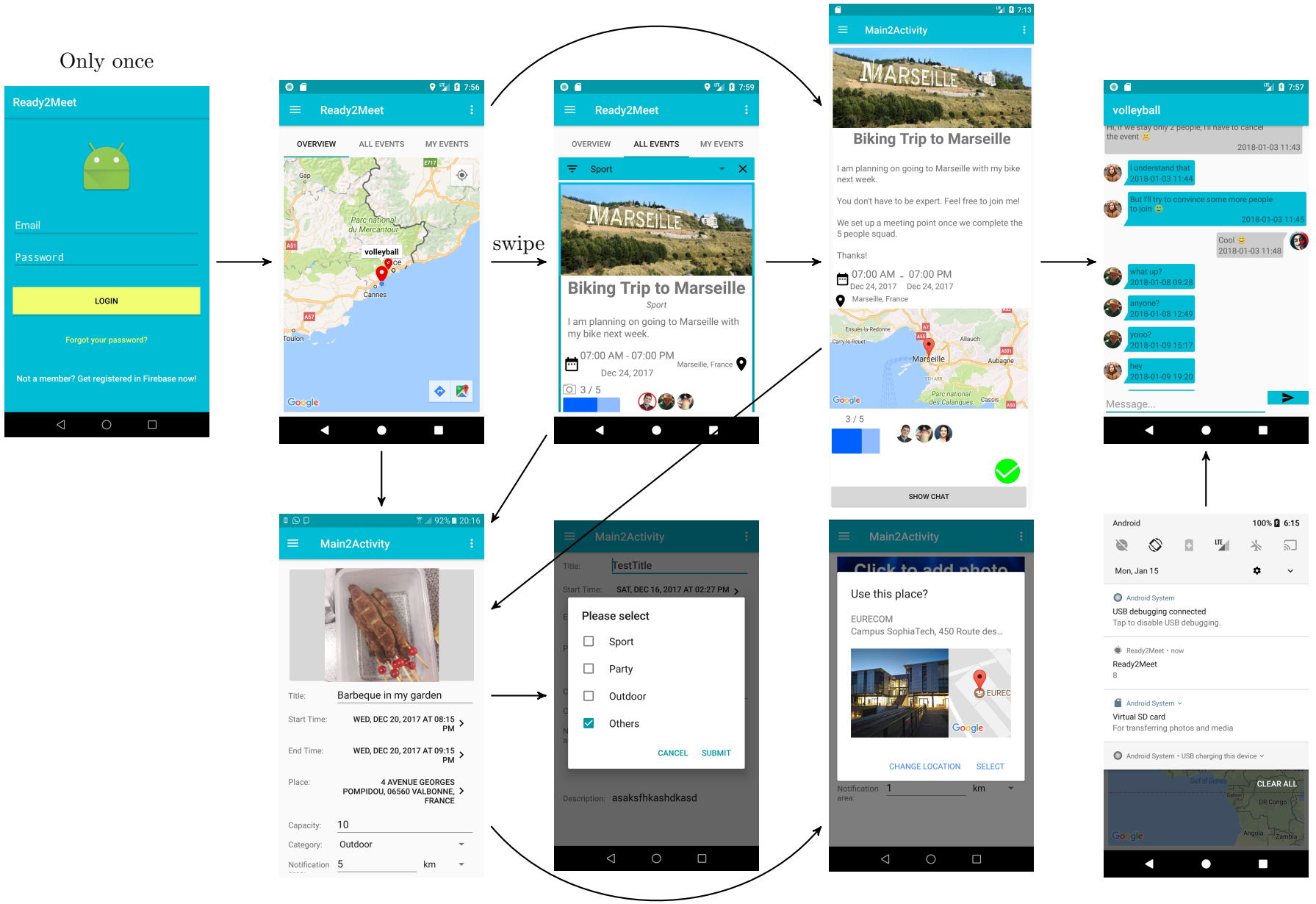

Figure 1: Click stream of the app

to show only events which are in his/her surrounding. To indicate the popularity of each event, the size of the markers of every event is scaled according to the number of participants of the event in relation to the number of participants for the most popular event. By clicking on the marker, the title of the event is shown. After clicking on the title, the event details are displayed. This page allows the user a quick orientation and aims at targeted presentation of events to the user.

By swiping to the right from the start screen, the events are listed with increasing event time. It is possible for the user to filter events according to predefined event categories. After filtering, the search is stored in shared preferences for later uses of the app. This allows us to reduce the user's clicks as we can assume that his/her preferences for event categories do not change frequently. By clicking on one of the events, its details are shown.

The event details show a complete overview of the event. On the top, every participant of an event can add photos to an image gallery. Afterwards, event details like a description, time and location as well as conflicts with the own calendar are given to all users of the app. Also, all participants are shown and a bar indicates how many slots have already been occupied. If the user participates on an event, he can be redirected to a chat.

A unique chat for every event can be used by all participants of an event to communicate before or after an event and thus discuss organizational details or share impressions. If new messages arrive in one of the user's chats, a notification is received also if the app is closed and serves as a quick link to the chat.

From every screen of the app (except the chat), the user can reach a screen to add new events with only one click. Several dialogs are used to make the creation of the event interactive and provide us with sanitized data.

6 Implementation

In this section, we discuss technical details of our implementation.

6.1 Database

To store our data (events, messages, users) and make them available to all users, we heavily make use of the features of Firebase[1] database. Firebase provides a real-time database and is therefore ideally suitable to make changes on data available to multiple users in real time. Apart from that, it can easily be integrated into Android apps (as well as iOS and web applications).

We need to store 3 different types of data structures namely user accounts, chat messages and events. For each of the structures, one java class defines the data fields which are stored. Class diagrams of the classes are shown in Figure 2.

[1] https://firebase.google.com/

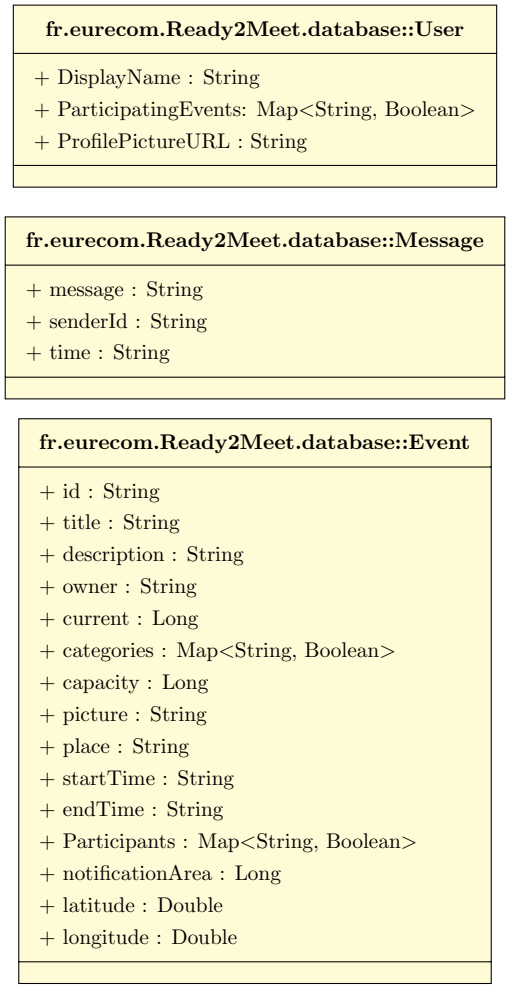

Figure 2: UML class diagrams of the database classes

6.2 Application Components

To provide the functionality, we implemented several different components which we discuss in this section.

The first crucial components of any Android app are the Activities shown in Figure 3. First, the three activities `ResetPasswordActivity`, `SignupActivity`, `LoginActivity` are responsible for managing user accounts and the corresponding login credentials. The class `ChatActivity` shows the chatroom for a selected event and requires the ID of the event in an extra in order to retrieve the chat messages. `AddEventActivity` is used to add one new event to the event database. The `MainActivity` shows the start screen of the app and displays all events on a map as well as in a list.

To make our app more interactive, we have a couple of fragments. When the user starts the app, the `DashboardFragment` is shown which provides an overview over all events in a map. Each event has an own marker on the map. Different sizes of the markers are

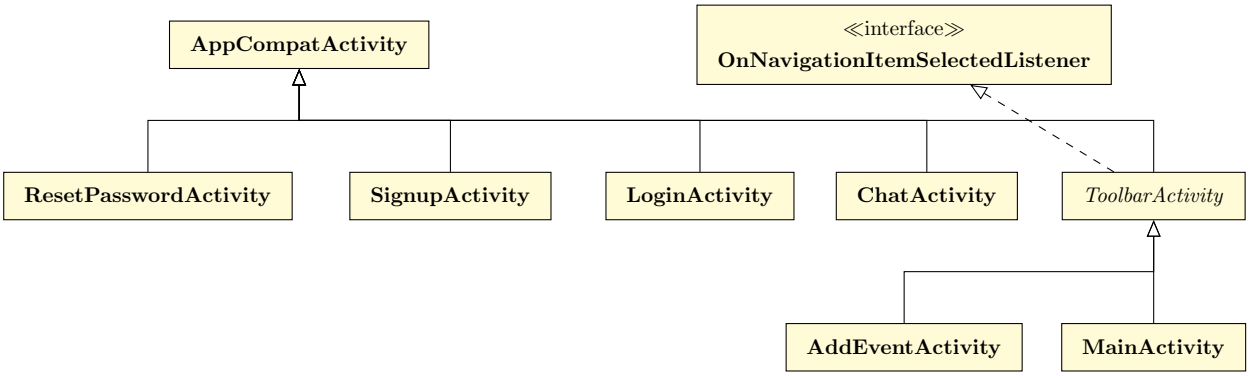

Figure 3: UML class hierarchy Activities

used to resemble the number of users of the corresponding events and thus allow the user to easily find more popular events. Next, the fragment `AllEvents` shows the events in a list and allows filtering events with respect to their category (e.g. Sport, Outdoor, Party, ...). After clicking on an event in one of the two views, the `EventDetailFragment` is displayed which makes the full event details available to the users, including an image gallery, description, time and place of the event. Furthermore, possible collisions of the event with the calendar are checked to give the user feedback if it is likely for him to be available during the event. Also, the user can join and unjoin an event and access the chatroom if he registered for the event.

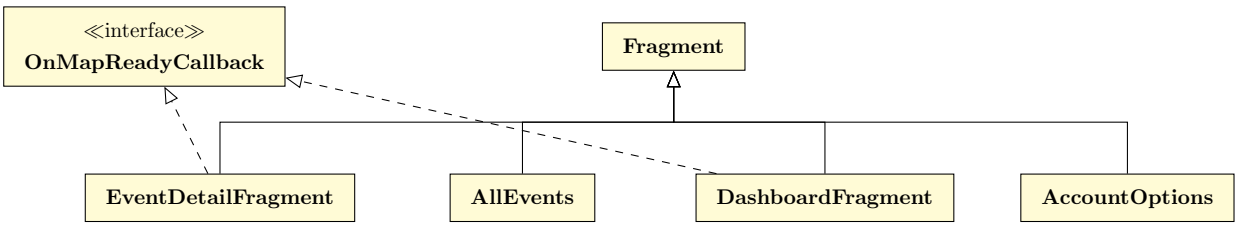

Figure 4: UML class hierarchy Fragments

Two services are continuously running to give the user feedback on changes in the Firebase database. `MyFirebaseInstanceIdService` is used to keep the users' ID which is necessary for Firebase Cloud Messaging (FCM)[2] up-to-date. The `ChatNotificationService` receives all notification messages from our FCM instance and displays them to the user. Like this, the user is always kept informed about new messages in the chats he joined.

6.3 Frontend/Backend Interaction

Figure 6 shows how the Android app interacts with the different backend components. Our backend completely relies on Firebase and makes use of multiple of its different features.

[2] https://firebase.google.com/docs/cloud-messaging/

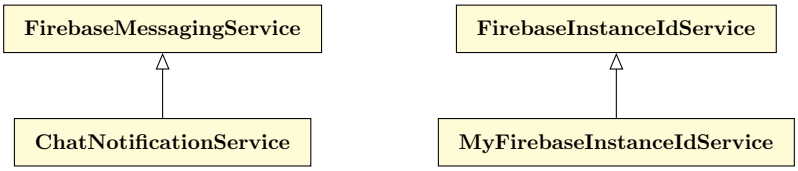

Figure 5: UML class hierarchy Services

First, the user authentication is based on Firebase's authentication features. This eases registering users, the log-in as well as advanced features like resetting passwords. We store additional data in the Firebase database to better address our users. Finally, every user has an own picture which is stored in Firebase storage.

In order to add event, we first make Firebase DB create a new ID for an event in the DB. For this event ID, the event data (i.e., title, description, time and place) are stored under the event ID. Finally, a folder for the event is created in Firebase storage and the event picture is uploaded to the folder. When additional pictures are added, they are stored in this folder and a value containing a link to download the image is added to the event instance in the DB.

If a user joins an event, he fetches the participants of the event to join from the DB and sends an updated participant list to the DB.

Finally, every user can write messages to a chat for each event he participates in. As we want to provide targeted push-notifications to users which are part of the chat, we implemented a Node.js component which is running as Firebase function instance (short function). The function is a listener to the DB and is notified whenever a new message is written to the chat. The new message is sent from the DB to the function which then retrieves the users of the corresponding chat and finally assembles a notification which is sent to all of these users. A service of the app receives the notification and displays the notification on the screen.

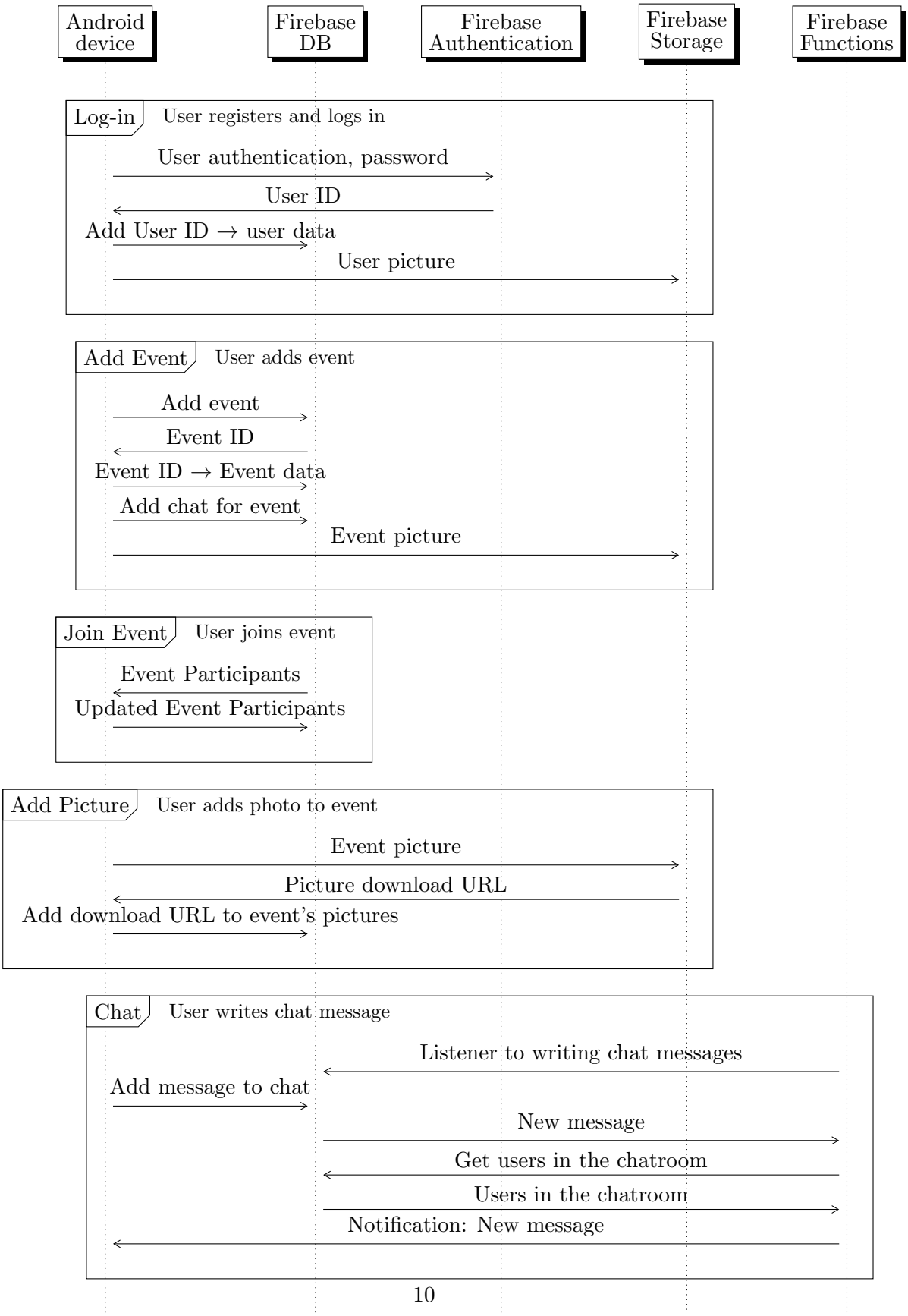

Figure 6: Component Interaction