Android Chat app using Firebase and Material Design

Matching process

1 Present potential matches
Find **list of profiles matching** the user's location and following search parameters. One challenge on top of this is **liked** and **disliked** flagged users should be filtered out. Probably need to use Geohash with 4 characters precisions with adjacent cells (~40 x ~20km each cell), that means 9 records.

```
{
 "searching_users": {
  "age": {
   "maximum": 28,
   "minimum": 18
  },
  "distance_max": 1000,
  "genders": {
   "female": true,
   "male": true
  }
 }
}
```

2 User actions
The user can use following actions; **like** or **dislike** on the profile once.
3 Reciprocal Matching
Identify **reciprocal likes** as a friend match.
4 Register the friend
Reciprocally **add the friend** to users for easy lookup later.

1. Configuration `/configuration`

Static client configuration files to so changes can be changed on the fly, if needed

Structure

```
{
 "configuration": {
  "chat": {
   "headers": {
    "string_max": 1000
```

```
  },
  "messages": {
    "string_max": 1000
  }
},
"matches": {
  "records_page": 20
},
"profiles": {
  "bio": {
    "string_max": 20,
    "string_min": 0
  },
  "images_max": 4,
  "name": {
    "string_max": 20,
    "string_min": 3
  },
  "upload_image": {
    "_comment": "Restrictions on image uploads",
    "bytes_max": 3000000,
    "height_max": 1000,
    "length_max": 2000,
    "bucket": "your-bucket-id"
  }
```

```
    }
  }
}
```

Security Rule

- Read: Anyone
- Write: only System Administrator

```
{
  "configuration": {
    ".read": true,
    ".write": false
  }
}
```

2. Profile `/profiles/$user`

Personal information of each user.

Structure

- `$user`: UID of user
- Depends on how the client use `avatar/coverPicture` against *profiles*, we can consider to separate them to new tables.

- All images will stored in Firebase storage, under

  `/storage/your-bucket-id/$user`

```
{
 "profiles": {
  "$user": {
   "public": {
     "name": "User name",
     "yearOfBirth": 1970,
     "gender": "male",
     "avatar": {
      "standard": "$storageStandard",
      "thumb": "$storageThumb"
     },
     "coverPicture": {
      "standard": "$storageStandard",
      "thumb": "$storageThumb"
     },
     "bio": "Some information",
     "interests": "some, interesting, things",
     "createdAt": 1473414120,
     "updatedAt": 1473414120
    },
    "private": {
     "dob": "1970-01-01",
```

```
      "createdAt": 1473414120,
      "updatedAt": 1473414120
    }
  }
 }
}
```

Security Rule

- Read:
  - Public: every one
  - Private: Owned resource
- Write
  - Logged in
  - Must owned resource

```
{
 "profiles": {
  "$user": {
   ".write": "auth != null && auth.uid = $user",
   "public": {
    ".read": true
   },
   "private": {
    ".read": "auth != null && auth.uid = $user"
   }
```

```
    }
  }
}
```

3. Match Settings

/searchingUsers/$user

Structure

```
{
  "searchingUsers": {
    "$user": {
      "age": {
        "maximum": 28,
        "minimum": 18
      },
      "distance_max": 1000,
      "genders": {
        "female": true,
        "male": true
      }
```

```
    }
  }
}
```

Security Rule

```
{
  "searchingUsers": {
    ".read": "auth != null && auth.uid = $user",
    ".write": "auth != null && auth.uid = $user"
  }
}
```

4. gridPictures `/gridPictures/$user`

Structure

- For easy handle number of images, I recommended to use image object keys as number from 1 to LIMIT (5 at the moment)

```
{
  "gridPictures": {
    "$user": {
      "1": {
        "standard": "$storageStandard",
        "thumb": "$storageThumb",
```

```
      "order": 1
    },
    "2": {
      "standard": "$storageStandard",
      "thumb": "$storageThumb",
      "order": 3
    },
    "3": {
      "standard": "$storageStandard",
      "thumb": "$storageThumb",
      "order": 2
    },
    "4": {
      "standard": "$storageStandard",
      "thumb": "$storageThumb",
      "order": 4
    },
    "5": {
      "standard": "$storageStandard",
      "thumb": "$storageThumb",
      "order": 5
    }
  }
}
```

```
}
```

Security Rule

```
{
 "gridPictures": {
  "$user": {
   ".read": true,
   ".write": "auth != null && auth.uid == $user",
   "$gridPictureId": {
    // This approach is must for performance
    ".validate": "$gridPictureId > 0 && $gridPictureId < 6"
   }
  }
 }
}
```

5. Like `/likes` & Dislikes `/dislikes`

Structure

```
{
 "likes": {
  "$like": {
   "userId": "google:123456789",
```

```
    "potentialId": "google:987654321",
    "createdAt": 1473414120
   }
  },
  "dislikes": {
   "$dislike": {
    "userId": "google:123456789",
    "potentialId": "google:987654321",
    "createdAt": 1473414120
   }
  }
}
```

Security Rule

```
{
  "likes": {
    ".read": true,
    ".write": "auth != null"
  },
  "dislikes": {
    ".read": "auth != null && auth.uid === $user",
    ".write": "auth != null && auth.uid === $user"
  }
```

```
}
```

6. Reports `/reports`

Structure

```
{
  "reports": {
    "$report": {
      "userId": "google:123456789",
      "potentialId": "google:987654321",
      "reason": "A long reason here",
      "createdAt": 1473414120
    }
  }
}
```

Security Rule

- Read
  - Logged in
  - Must owned resource

```
{
  "reports": {
    ".read": "auth != null &&
root.child('reports').child($like).hasChild(auth.uid)",
```

```
    ".write": "auth != null"
  }
}
```

7. Channels `/channels/$user`

Structure

- Each conversion will create 2 channels
  - Sent: Owned by the one who start the conversion
  - Received: for who received the conversion
- Depend on `$channel.updatedAt` and `$message.updatedAt` we can know the opposite user read the message or not.
- Query with only `userId` can return list of conversions, and easilly order by `updatedAt`
- Using 2 different keys of user UUID make the security rules possible

```
{
  "channels": {
    "$channel": {
      "userId": "google:123456789",
      "potentialId": "google:987654321",
      "conventionKind": "Sent",
      "createdAt": 1473414120,
      "updatedAt": 1473414120
    },
```

```
    "$channel": {
      "userId": "google:987654321",
      "potentialId": "google:123456789",
      "conventionKind": "Received",
      "createdAt": 1473414120,
      "updatedAt": 1473414120
    }
  }
}
```

Security Rule

- Read
  - auth.uid match with userId or potentialId
- Write
  - auth.uid match with userId

```
{
  "channels": {
    "$channel": {
      ".read": "auth.uid == root.child('/channels' + $channel).userId || auth.uid
== root.child('/channels' + $channel).potentialId",
      ".write": "auth.uid == root.child('/channels' + $channel).userId"
    }
  }
```

```
}
```

8. Channel meta data

/channel-headers/$user/$potentialUser

Structure

- Can access the conversion via REST url

  `/channel-headers/user-uuid-who-sent/user-uuid-who-received`

```
{
 "channel-headers": {
  "$user": {
   "$potentialUser": {
    "alex": "is a hero!"
   }
  }
 }
}
```

Security Rule

- Read

    - `auth.uid` match with `$user` (Sent) or match with `$potentialUser` (Received)
- Write
    - `auth.uid` match with `$user` (Sent) or match with `$potentialUser` (Received)

```
{
 "channel-headers": {
  ".write": true,
  "$user": {
   ".write": true,
   "$potentialUser": {
    ".read": "auth != null && (auth.uid == $user || auth.uid == $potentialUser)",
    ".write": "auth != null && (auth.uid == $user || auth.uid == $potentialUser)"
   }
  }
 }
}
```

9. Conversations `/conversations/$user/$potentialUser`

Structure

- Can access the conversion via REST url

  `/conversations/user-uuid-who-sent/user-uuid-who-received`

```
{
 "conversations": {
  "$user": {
   "$potentialUser": {
    "$message": {
     "message": "I would like to talk with you",
     "createdAt": 1473414120,
     "updatedAt": 1473414120
    }
   }
  }
 }
}
```

Security Rule

- Read
  - `auth.uid` match with `$user` (Sent) or match with `$potentialUser` (Received)
- Write
  - `auth.uid` match with `$user` (Sent) or match with `$potentialUser` (Received)

```
{
  "conversations": {
    ".write": true,
    "$user": {
      ".write": true,
      "$potentialUser": {
        ".read": "auth != null && (auth.uid == $user || auth.uid ==
$potentialUser)",
        ".write": "auth != null && (auth.uid == $user || auth.uid ==
$potentialUser)"
      }
    }
  }
}
```

10. Friends `/friends/$user`

Structure

```
{
  "friends": {
    "$user": {
      "$potentialUser1": true,
      "$potentialUser2": true
    }
```

```
  }
}
```

Security Rule

```
{
  "friends": {
    "$user": {
      ".read": "auth != null && auth.uid == $user",
      ".write": "auth != null && auth.uid == $user"
    }
  }
}
```

11. Matched Users

/matchedUsers/$user

- Is list of matched potential Users base on *searchingUsers* conditions
- Will be return from GAE server though a search engine (ElasticSearch)

```
{
  "matchedUsers": {
    "$user": {
      "potentialUser1": true,
```

```
    "potentialUser2": true
   }
  }
}
```

12. User location

/userLocations/$user

Structure

```
{
 "userLocations": {
  "$user": {
    "$userLocation": {
      "lat": 123.4567,
      "long": 765.4321,
      "createdAt": 1234567890,
    }
  }
 }
}
```

Security Rule

- Write

    - User logged in
- Read: false (only system)

```
{
  "userLocations": {
    "$user": {
      ".read": false,
      ".write": "auth != null && auth.uid == $user"
    }
  }
}
```