

System design document for AWME bokbytarapp

Ali Alladin, Magnus Andersson, William Hugo, Elias Johansson

2020-10-23

1.0

1 Introduction	1
1.1 Definitions, acronyms, and abbreviations	1
2 System architecture	1
3 System design	3
4 Persistent data management	5
5 Quality	6
5.1 Access control and security	6
6 Potential improvements	7
7 References	7
8 Appendices	8
8.1 Login	8
8.2 Register	9
8.3 SearchPage	10
8.4 ListingPage	11
8.5 ChatRoomsPage	12
8.6 MessageList	13
8.7 AccountPage	14
8.8 AddListing	15
8.9 EditListing	16
8.10 Settings	17

1 Introduction

This document is intended to present our system design for our application “Bokbytarapp”. The program functions as an online marketplace where users can sell used items that relate to their education in some form. No transactions are handled in the app as it only functions as a mediator. The program needs a minimum version of Android 7.0 to be installed and works on phones and tablets.

1.1 Definitions, acronyms, and abbreviations

Listing: Refers to all the items that are being sold in the app.

Activity: Activity is a class in Android that takes care of creating a full screen window for you in which you can place your UI components.

Firebase: Firebase is a platform developed by Google for creating mobile and web applications. The functions from Firebase used by us are Database and Authentication.

RSA cryptography: Rivest–Shamir–Adleman cryptography. A kind of asymmetric cryptography, meaning that there’s two keys, one public and one private, used for encrypting and decrypting strings. The private key cannot be derived from the public key, meaning that anyone can encrypt messages for a specific user using that user's public key, but only the user the message is for can decrypt the message using their private key.

Design Pattern: A typical solution to a common software design problem.

2 System architecture

When you start the application you have to login by entering Email and Password in the Login activity (See [8.1 Login](#)). The credentials are then checked by Firebase Authentication. If the account exists and the credentials are correct it will start a websocket session with our database, which will persist throughout the whole run of the application and allow for real time updates to the items shown in the application, and messages sent and received.

If you don’t have an account you have the possibility to register one by clicking on the register button found just beneath the login button. This takes you to the Register activity (see [8.2 Register](#)) which mainly consists of input fields where the user fills in their information. There are multiple conditions that need to be met before the account is registered. If the conditions are not met an error message is displayed with more information. When all the fields are filled and conditions are met the user is presented with a confirmation message and returned to the Login activity.

After you login you come to the MainActivity. MainActivity has a bottomNavigationView that the user can use to navigate between the three pages which are fragments. The first page that the

user meets in MainActivity when he logs in is SearchPage (see [8.3 SearchPage](#)) where all listings available in the database are displayed as cards, should another sale be added it will update on this screen. There is a search bar on top of this page where the user can search among all listings. Next to the search bar there is a button that opens up a popup menu. The listings on the SearchPage can be sorted in desired order based on the item-selected in the popup menu.

A listing (card) can be interacted with, which will take you to ListingPage (see [8.4 ListingPage](#)) where you will see the full information about the listing. Based on whether the listing is uploaded by yourself or another user the ListingPage has some differences. If it's someone else's listing there's a button to save the listing as a "favourite". Further there's information about the seller at the bottom of the ListingPage. It's also from here you can contact the seller, but more about that will be mentioned later. If the listing on the other hand is your own it doesn't have the option to save the listing. Instead of the seller information there are two buttons displayed at the bottom, one to edit the listing that opens EditListing, and the other is to delete the listing.

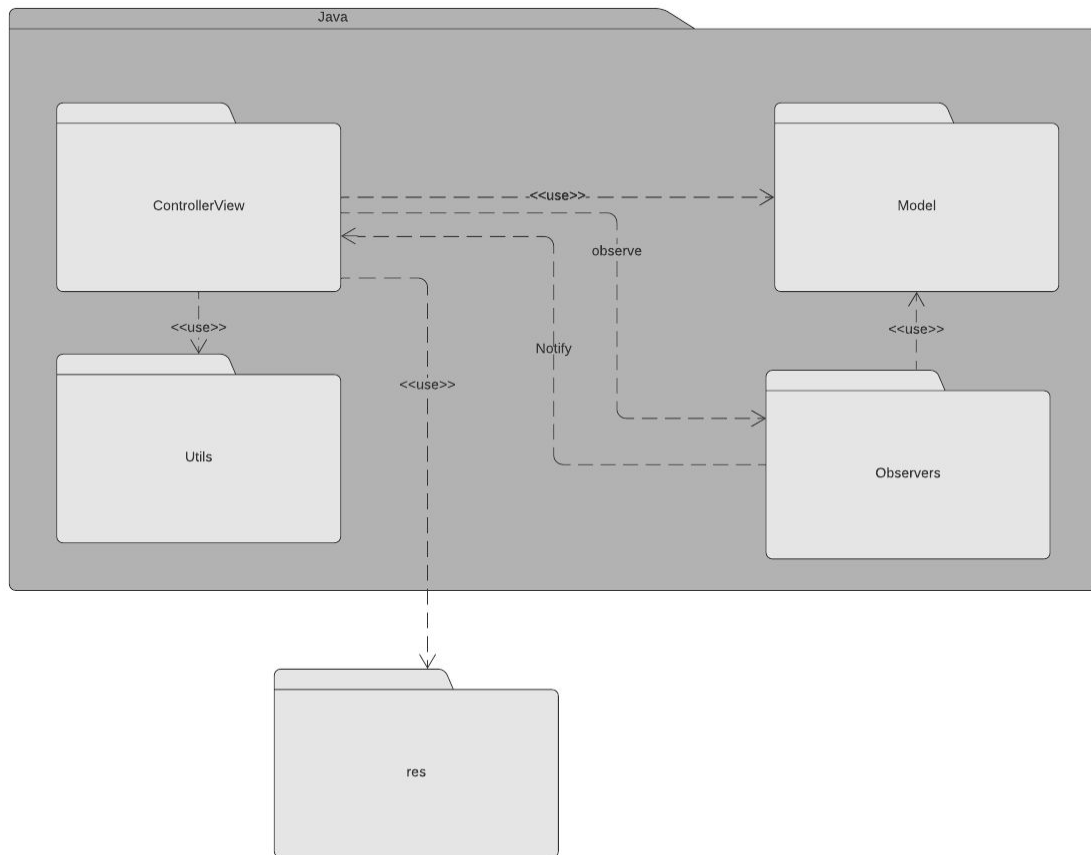
Back in the MainActivity the second page after the SearchPage that can be accessed with the BottomNavigationBar is MessagePage (see [8.5 MessagesPage](#)). On this page you can see all users you have contacted or been contacted by. The conversation holds information about the most recent message, time sent, profile picture and name. This functionality comes from ChatroomAdapter and MessagePage. Interacting with a user-card here takes you to a full log of messages, MessageListActivity (see [8.6 MessageListActivity](#)). Here you'll be able to send messages that will be encrypted with RSA-encryption by the cryptography-related classes. It will then be sent to two chat rooms in the database so both users can decrypt it.

On the third and final page, which is AccountPage (see [8.7 AccountPage](#)) you can see your own listings, see saved listings, post a new listing or go to settings. Both the lists on AccountPage contain the same ListingCards as SearchPage and therefore also take you to ListingPage. A new listing can be added by clicking on the big orange "Ny annons" button. This will take you to the AddListing activity (see [8.8 AddListing](#)) where a new listing can be uploaded by filling all the fields of the form. Just like the register "form" there are some conditions that need to be met before the listing can be added. You will be informed if a condition isn't met. When everything is okay and the listing is successfully added a confirmation message is displayed and the user is redirected back to the AccountPage. The EditListing activity (see [8.9 EditListing](#)) that was mentioned earlier is almost identical to AddListing with the only difference being that the EditListing has all the fields filled.

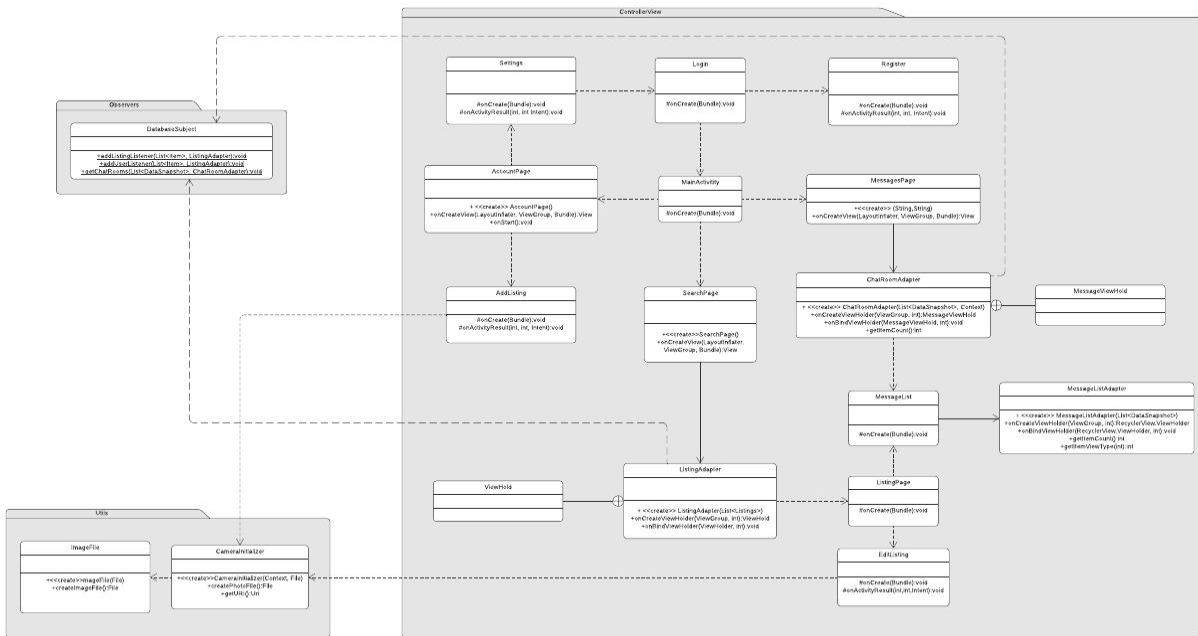
If one clicks on the "settings-button" the settings page (see [8.10 SettingsPage](#)) appears. On this page the user's can change their own information and also have the option to log-out from the application. The changes made in this page are saved in the database and in Firebase Authentication.

If you are to close the application at any stage, you would break the connection to the database, and be greeted with the login screen upon the next launch.

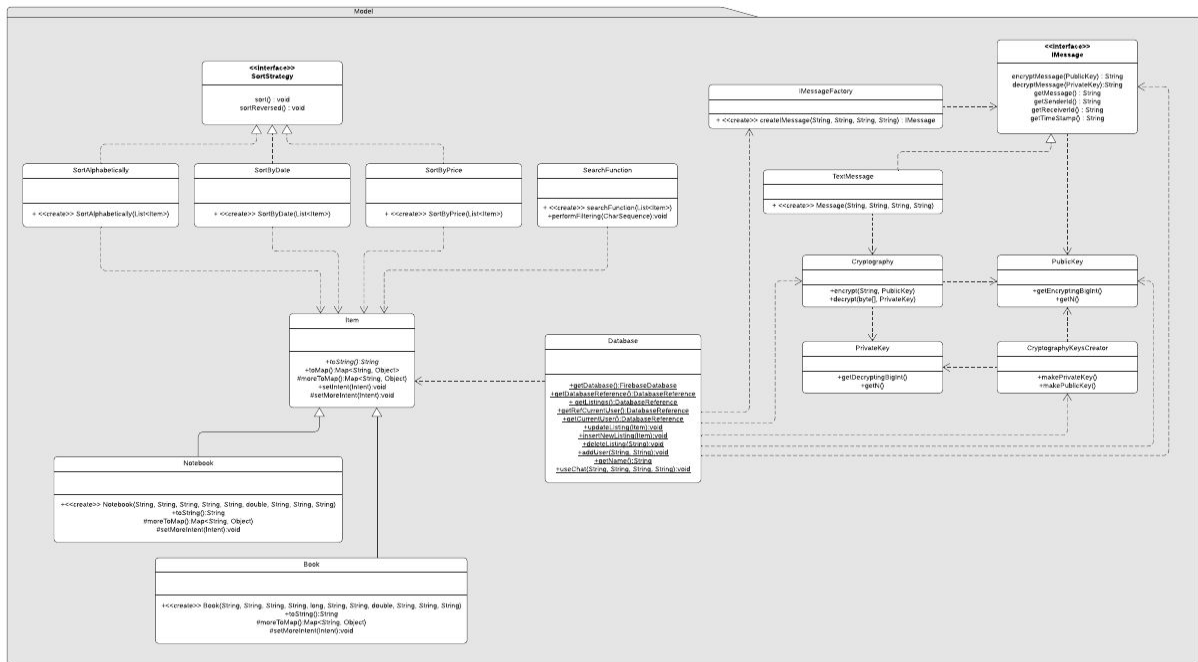
3 System design



Package diagram



Class diagram of ControllerView and Utils packages



Design Model

The MVC pattern is used by dividing our program into three packages: Model, ControllerView and res. Model represent all the model classes in our program. ControllerView are classes that function to some extent as classes that create views and ordinary controllers. This is because

the Controller and View classes are hard to separate in Android. In a sense the package res is an extension of the view. Res contains the xml files and images in our project. There is also an additional package called utils that holds two classes that interact with the device's camera and is used by ControllerView. There is an additional package called observers which handle some communication between the database and ControllerView to avoid certain dependencies. ControllerView has adapter classes that use the model to update it and retrieve data from it. This is always done on the highest abstraction level.

We did not find a good way to represent our Res folders as a class diagram.

Module design pattern in form of MVC.

We make use of the **Template method** design pattern in our item hierarchy, because of methods with a majority of overlapping behaviour, but where some deviations are guaranteed. For communicating with our database we have implemented an **Observer** pattern where the classes ListingAdapter and ChatroomAdapter observes the DatabaseSubject class. This has another benefit of containing our database, firebase, specific imports to the Database.

Factory-like pattern where IMessageFactory hides creation logic for IMessages, even though there will only be one type in this prototype, and generalizes the creation process.

Our sorting is structured with a **Strategy pattern** to follow the open/closed principle. New ways to sort our listings are almost guaranteed to come along as the application is extended, and this pattern is designed with that in mind.

Singleton - the Database class is a singleton since it is a shared resource used by several different parts in the program.

The domain model contains some conceptual objects that are not represented in our design model. This is because some objects like User or Chatroom are stored in our database. Other objects are represented in the design model such as Cryptography, Sort and Message but their technical implementation is not shown in the domain. There is also the case when two entities are shown as one in the domain. This is the case with the database, which has two classes related to it but it should be seen as an aggregate of both the classes and the database that is used.

4 Persistent data management

We use a database to store all items, messages and users in our program. Each item is stored as a dictionary of strings and numbers.

Books isbn numbers have to be stored as long integers because they are 13 digits long.

We store the price as doubles, to retain the decimals.

Images are stored as encrypted Base64 strings.

The encrypted messages are also stored as Base64 strings in the database.

Users are stored as a unique value generated by our database upon creation of the user, with attached name, and two lists of items, one that the user is selling, and one that the user favours.

Some icons are stored in our res folder.

5 Quality

We write jUnit tests, which gradle will run every time you build the application, and travis, the continuous integration service, will run every time someone submits code to our repo. These tests can be found at

`\bokbytarapp\app\src\androidTest\java\com\example\amwe` and

`\bokbytarapp\app\src\test\java\com\example\amwe`. AndroidTest can not run with coverage.

The database related classes are not tested directly but indirectly with other tests. It would not be that useful to write a Mock-Database to see if objects are correctly uploaded and retrieved.

Here is a link to our Continuous integration service, Travis,

<https://travis-ci.com/github/magnusGU/AMWE>

Below is a list of all known issues

- Messages are not properly shown sometimes if emojis are involved.
- Messages take a while to load in when you scroll fast in a chatroom.
- Application crashes if one tries to back out from the phone's gallery when choosing profile picture.
- Photo gets rotated on some devices when it's taken through AddListing and EditListing.

Gradle handles both quality and dependency checks and reports. The following is a link to a gradle generated report, containing results of building, checking and generating a dependency tree. For the most in detail report check under the console log tab, where the whole "gradlew clean build app:dependencies" command can be followed step-for-step.

<https://scans.gradle.com/s/lgawvoymi5okc>

5.1 Access control and security

To use our application you have to have a user account, or create one with a "semi-valid" email, (no verification mail is sent to the email, so does not have to be real). But there has to be an at-symbol '@' followed by some service dot some domain I.e name@mail.com.

These accounts, and their password security is handled by our database, Firebase.

6 Potential improvements

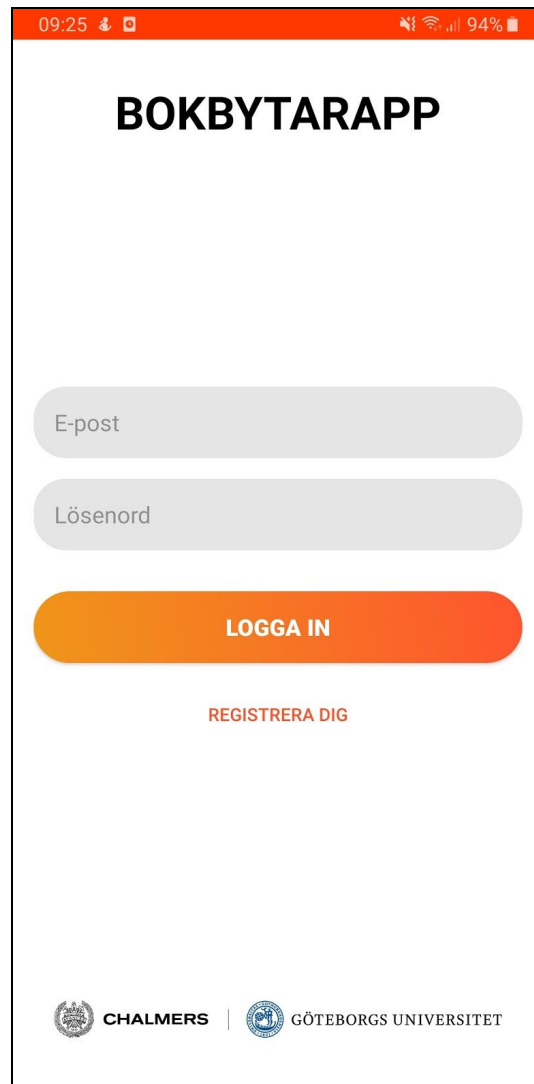
- Performance issues is something that would need to be addressed if the product would be a commercial one. Sometimes it takes time to load the GUI.
- In a commercial product the user should remain logged in after first time application usage and login.
- Messages should be shown in order and be marked as read when you read them. Store messages when you have downloaded them, store them instead of downloading them everytime.
- Messages can't be deleted right now. This would of course not be
- Settings could be improved a lot. E.g. It's currently very easy to change the password which is unsafe. Further it would be good if there was an option to delete the account.
- The handling of the database is sometimes not ideal. In several cases too much data is downloaded even though little of it is needed. A solution would be to refactor our database structure.

7 References

- [Android](#)
- [Firebase](#)
- [Gradle](#)
- [Travis](#)
- [JUnit 4](#)
- [RSA cryptography](#)

8 Appendices

8.1 Login



The screenshot shows a mobile application interface for 'BOKBYTARAPP'. At the top, there is an orange status bar with the time '09:25', signal strength, Wi-Fi, and battery level '94%'. Below the status bar, the app title 'BOKBYTARAPP' is displayed in bold black text. The main content area contains two light gray rounded rectangular input fields: the first is labeled 'E-post' and the second is labeled 'Lösenord'. Below these fields is a large orange rounded rectangular button with the text 'LOGGA IN' in white. Underneath the button is the text 'REGISTRERA DIG' in red. At the bottom of the screen, there are two logos: the Chalmers logo on the left and the Göteborgs Universitet logo on the right, separated by a vertical line.

09:25 94%



BOKBYTARAPP

E-post

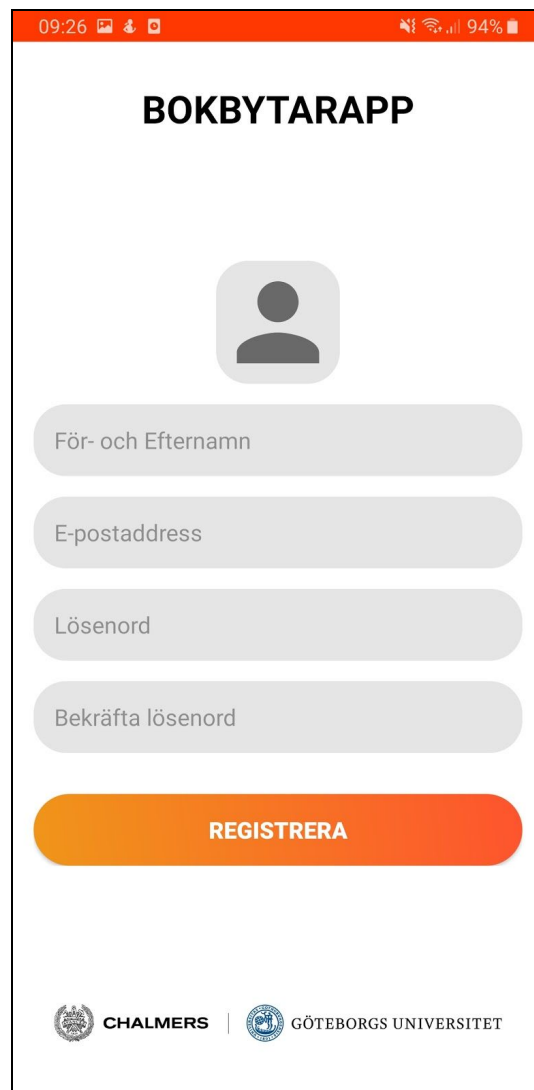
Lösenord

LOGGA IN

REGISTRERA DIG

 CHALMERS |  GÖTEBORGS UNIVERSITET


8.2 Register



The image shows a mobile application interface for registration. At the top, there is an orange status bar with the time 09:26, signal strength, and battery level at 94%. Below this, the app title "BOKBYTARAPP" is centered in bold black text. Under the title is a grey rounded square containing a black silhouette of a person's head and shoulders. Below the profile icon are four light grey rounded rectangular input fields, each with a placeholder text: "För- och Efternamn", "E-postadress", "Lösenord", and "Bekräfta lösenord". Below these fields is a prominent orange rounded rectangular button with the white text "REGISTRERA". At the bottom of the screen, there are two logos side-by-side: the Chalmers logo (a circular emblem) followed by the text "CHALMERS", and the Göteborgs Universitet logo (a circular emblem) followed by the text "GÖTEBORGS UNIVERSITET".

09:26 94%

BOKBYTARAPP





För- och Efternamn

E-postadress

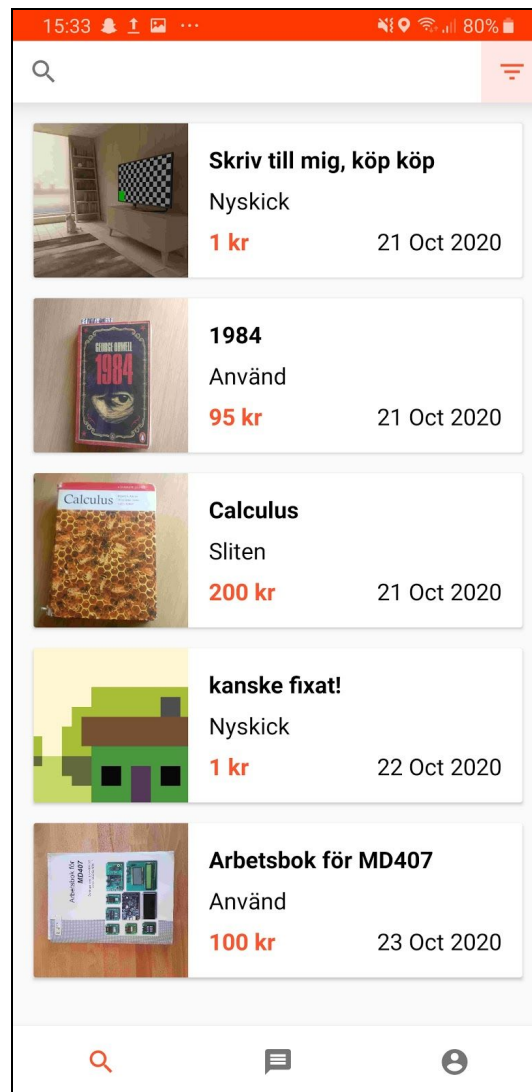
Lösenord

Bekräfta lösenord

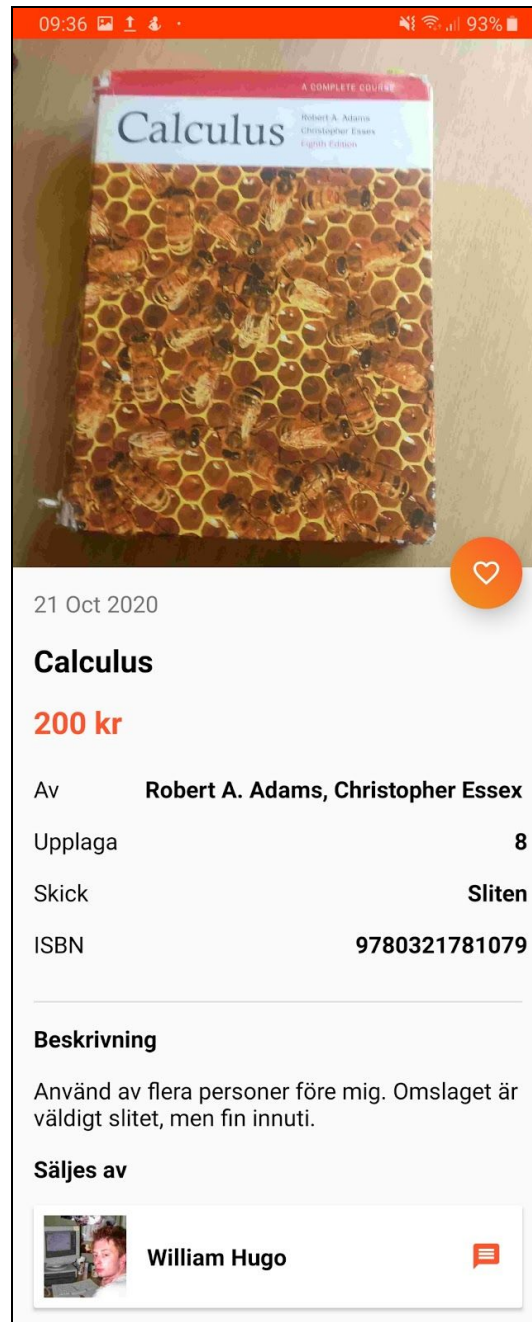
REGISTRERA

 **CHALMERS** |  **GÖTEBORGS UNIVERSITET**

8.3 SearchPage



8.4 ListingPage



8.5 ChatRoomsPage



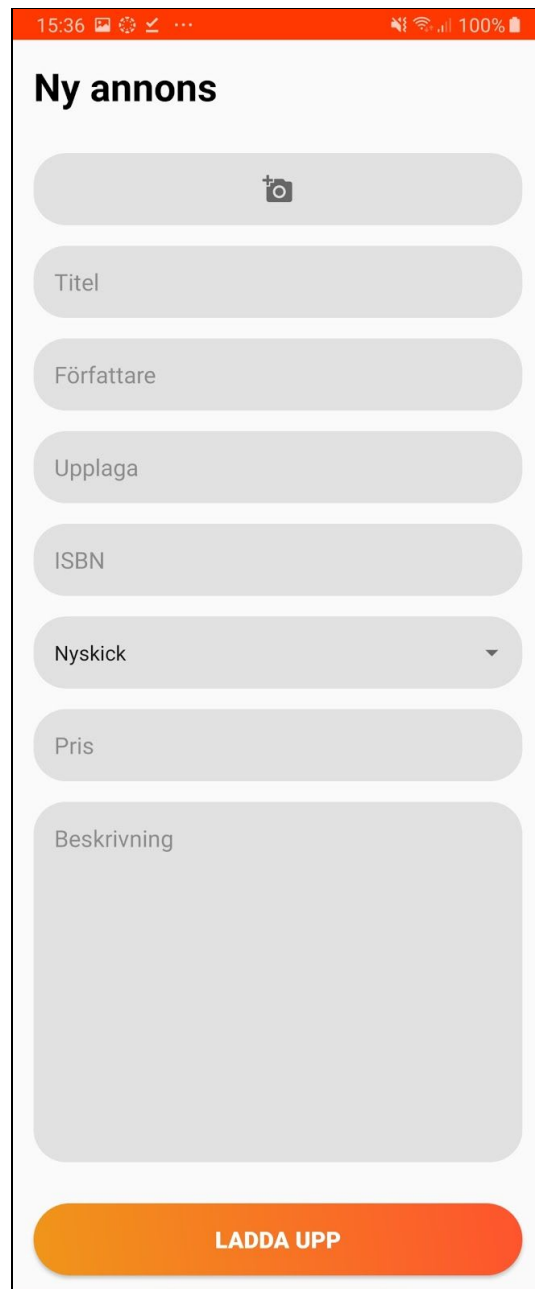
8.6 MessageList



8.7 AccountPage




8.8 AddListing



A mobile application interface for adding a new listing. The screen has an orange header bar with the time 15:36, signal strength, and 100% battery. The title 'Ny annons' is in bold black text. Below the title are several input fields: a photo upload button with a camera icon, text fields for 'Titel', 'Författare', 'Upplaga', and 'ISBN', a dropdown menu for 'Nyskick', and a text field for 'Pris'. A large text area for 'Beskrivning' is below these. At the bottom is a large orange button labeled 'LADDA UPP'.

Ny annons



Titel

Författare

Upplaga

ISBN

Nyskick ▼

Pris

Beskrivning


LADDA UPP

8.9 EditListing

15:30

81%

Ändra annons



MD407
Övningar med ETERM8 och
GCC, GDB för ARM

Arbetsbok för MD407

Roger Johansson

5

9789189280298

Använd ▾

100.0

Har skrivit med blyerts.

SPARA

8.10 Settings

15:39

100%

Inställningar

Ali Alladin

ali.alladin@mail.com

Nytt lösenord

Bekräfta nytt lösenord

SPARA

LOGGA UT