Time Banking System

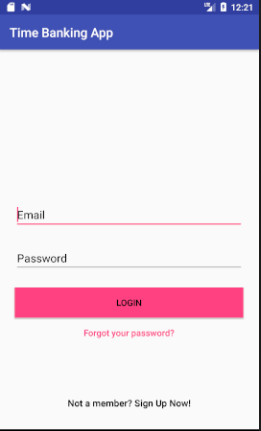
Software Design Walkthrough

Vincent Zhang, Cletus Andoh, Joseph Urie, Xin Tang

Table of Contents

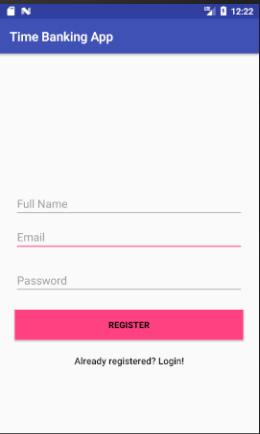
1. Login Page
2. Sign Up Page
3. Main Page
4. Description Page
5. Adding Jobs Page
6. My Jobs Page
7. Timer Page
8. Profile Page

1. Login Page



When the app is started, users will be presented with a login screen. From here, users can choose to input their credentials and login via email and password, or they can click on the sign up button to go to the registration page. If login is successful, the user will be directed to the main page. There are various tests such as making sure none of the fields are empty and also if the credentials are wrong, a message will be displayed to the user telling them to re-enter their credentials.

2. Sign Up Page



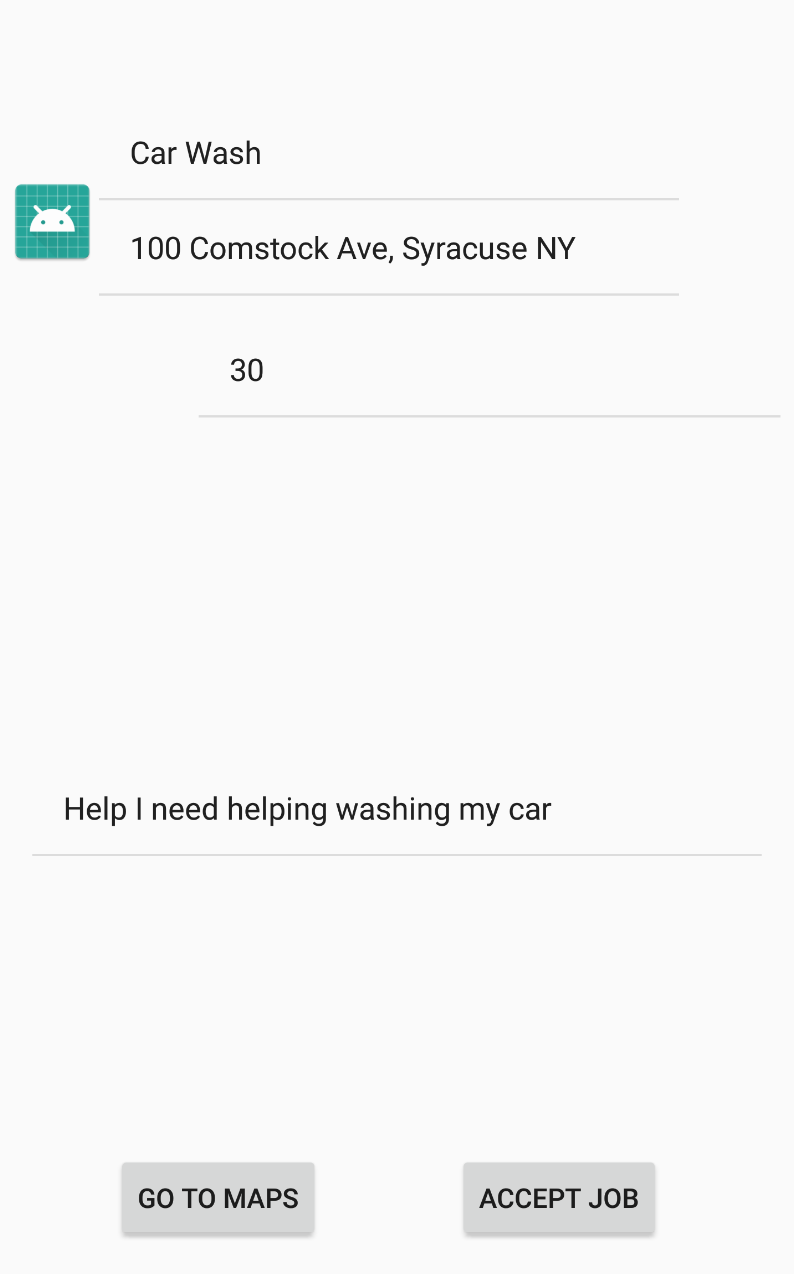
A new user of the application will need to click the sign up button at the login page to register for an account. On the sign up page, they will need to fill in their name, email, and password. There are tests implemented to make sure that all fields are filled and also the password is at least six characters long. Once they click the register button, the system will be interacting with Firebase authentication to check if there already is a user that is registered under that email. If there isn’t the user is created by using a Job object from the Job class which will contains information on the user and give the user thirty free minutes as well as a 0 for rating. That Job object will be stored in the Firebase Database under “users”. For the authentication page, when the user is created, the user will be assigned a unique Id for identification purposes for various other functions throughout the application.

3. Main Page



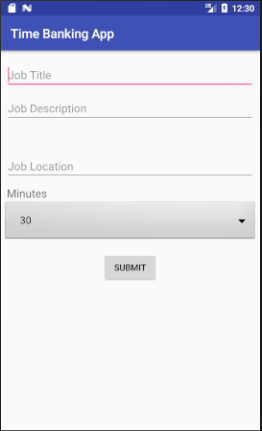
The main page has a simple layout design consisting of four buttons and a search bar. There is a “Profile” button which leads to the each users specific profile. The “My Jobs” button leads to a page containing all the jobs that a user has accepted. From there they can start the jobs they accepted. The “Add Job” button leads the user to a form where they can enter details about a job they want completed. The “Log Out” button simply logs out the user. The search bar is used to search for all jobs matching what the user has entered.

4. Description Page



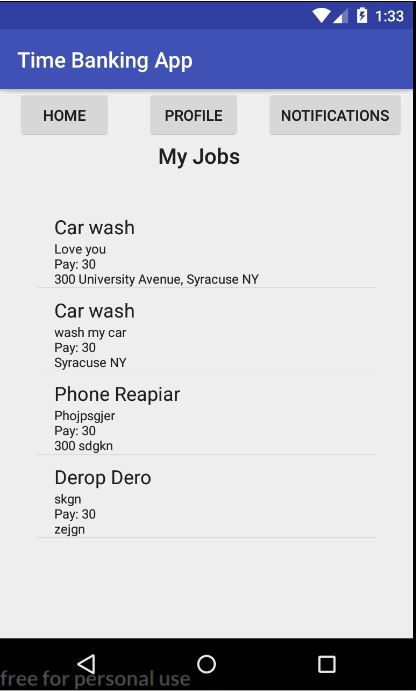
The job description page displays more information that the user wasn't able to see when they search for a job such as the job location and job pay. When a user searches for a job, they can only see the title and description to get a hint of what the job could possibly entail. But with this description page, it allows the user access to more information. Along with that, it includes two buttons. The “Go to Maps” button shows the user exactly where the job is taking place and can get directions to the location from their current location. Lastly, it includes an “Accept Job” button that lets the user accept the job they searched for.

5. Adding Jobs Page



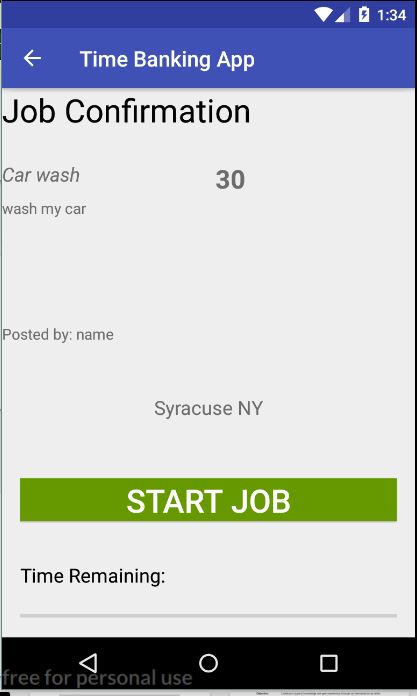
When the user decides to request a job, they can click on the add job button which will take them to a page where the user can input the job title, description, location, and scroll through a list of minutes they would like to request which is displayed in 30 minutes intervals. There are tests to make sure that all information are filled and also the system will need to check and make sure the user has enough minutes on their account to request the job. The Job object contains the minutes a user has on their account and also the available minutes as well because the currency won’t be exchanged until after the job is completed so we need to keep track of the available minutes the user has to avoid requesting too many jobs that exceed their minute balance. This is done by reading from the database and storing those values in a User object in which the minutes can be retrieved via setters and getters. If the user has enough minutes, then the Job object will be used to store the job information and store it in both the jobs folder on the database and also under offered jobs folder in the user that requested the job.

6. My Jobs Page



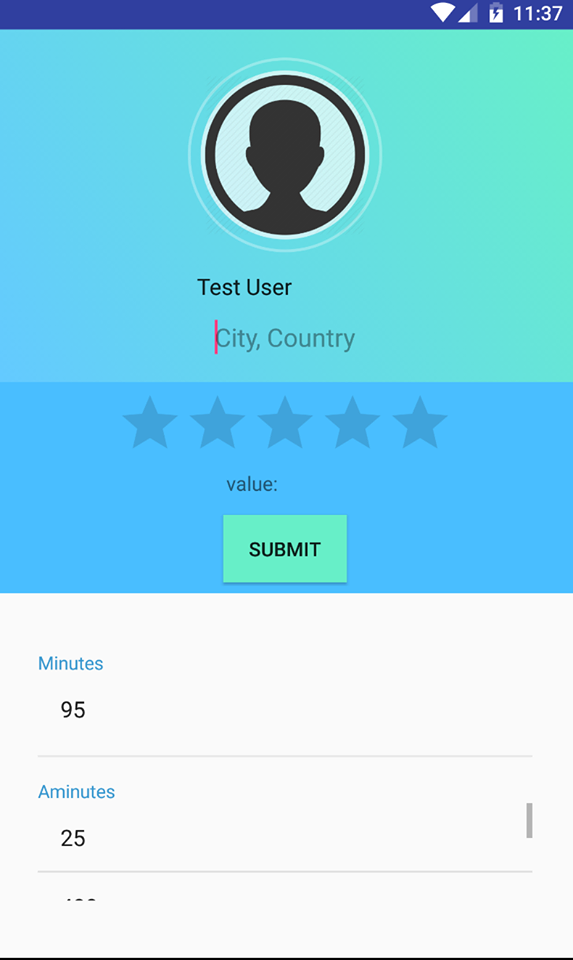
This page can be accessed from the user’s main page. It displays a listview that includes all of the jobs that the user has signed up for and needs to complete. It is dynamically updated to match the database in real time. When you click on an item within the list, you will be sent to the timer page. Each timer page is tailored to the same job you clicked on in the listview, each one being different. Also included our tab buttons to navigate to other pages within the app. The user must be signed-in to for this page to work as it is uniquely attached to their profile.

7. Timer Page



The timer page is the page where the job is confirmed one last time before it is completed by the user. After listing all the job information, the user can press the green start button which will begin a timer for their work to complete. Once the timer is finished, the job is marked finished, and removed from the database. When the timer ends, the actual transaction of the minutes occurs. The user who completed the job gains the designated amount of minutes while the account that posted the job has minutes removed. All minutes transactions are saved on our database. The timer page is modified by the listview to only display the information of the job that was selected on the previous page. To exit this page, hit the back button located on the top of the page and you will be returned to the My Jobs page where the listview has been updated to handle the removed job.

8. Profile Page



The Profile page is the page where user are able to edit their profile to other user. Using the firebase database we were able to retrieve data from database which is done in the login page and save into user information. The page collects name, minutes and aminutes. Furthermore the profile page was meant also meant to connect it to rating page however we came into a logical error. We didn’t account for when after a job is completed, how would responder connect to the job completed and rate the requester. Therefore we conclude the rating system would need to be trashed because it would pointless for responder to rate the requester.