Landlord-Tenant Communication App

Motivation:

The objective of the app was to facilitate and encourage communication between a landlord and all of his or her tenants. Many times, tenants find themselves in need of contacting their landlord, however, because of the inconveniences that doing so may create, they end up not communicating with their landlords to resolve their issues. In addition, many times for tenants, renewing a lease or contacting other individuals within the building are also inconvenient tasks. Certain apps may already exist for this case, but our implementation is unique in the aspect that it will allow for lease renewal, maintenance requests, as well as bill payment. These features are typically not a part of many of the apps that tout landlord and tenant communication.

Our Solution:

Our application aims to bridge the communication gap between landlords and their tenants. A tenant can very easily get into contact with their landlord (and vice versa) with the convenience of being able to do so at any time or place. The application provides both parties a method to communicate freely through messages or to send specific requests such as maintenance necessities or rent payments. The probable reasons for landlords and tenants to communicate have been categorized into 5 main categories: maintenance requests, utilities, bills, contact information, and lease renewals. For anything that doesn't fall into one of these categories, there is a messaging feature in the application that allows for anything else to be discussed. Some functions of the app as a tenant include being able to renew your release and sign on your phone, viewing a history of your bills, viewing your current bill, being able to pay your current rent through Venmo, sending maintenance requests to the landlord, and being able to view contact information of other tenants in the building if they choose to have their information available. As a landlord, a user is able to view all of their tenants as well as all information for each tenant, view the current rent due for each tenant as well as requesting rent payments, and view the maintenance requests and mark each as complete.

User Interface (UI):

The application opens to a login activity and depending on which type of user logs in, a different view is displayed. Both account types share a similar homescreen and the layouts of each are also very similar. The tenant interface provide buttons that are clearly labeled for quick access to important and common tasks such as viewing bills, paying rent, sending maintenance requests, or simply sending the landlord a message. Pressing any of these buttons will open up the respective activity from which the user can complete a task or return to the main menu. For the maintenance request activity, a user can give each request a title and description and after submitting, it will show up in the list of submitted requests on the activity. A similar interaction works with the utilities and messaging features. For the lease renewal, a tenant is able to sign his or her signature using their finger and when finished has the option to either clear the signature or submit it for the renewal upon agreeing to the terms and conditions. In the contacts activity, all contact information of landlords and other tenants in the building is available. The user has the option to enable or disable their own information from showing up in this list for others to see. Lastly, if a user clicks on the bill payment button, the app will first check to see if the Venmo app is installed on the phone. If it is, it will directly open to the venmo app. If it is not, then the app will open an in-app web browser which will load up the login page for Venmo.com. This enables the user to quickly and effortlessly pay their current rent dues to their landlord.

As a landlord, the interface is similar in that the home screen provides clearly labeled buttons for quick access to the most common features. The landlord is able to view all maintenance requests from tenants and mark each as complete, after which the request will be moved from the pending requests list to the completed requests list. In the contacts page, the landlord will be able to view all tenants regardless of their visibility preference (as that is only to hide information from other tenants) and unlike the tenant interface, the landlord has access to all tenant information such as address, last rent payment, current rent balance, and other information. The messages feature works the same for landlords as with tenants. It is a multi-user messaging feature in which a landlord communicates with all tenants within the chat so that every tenant is able to view the messages as well. This makes it easy for multiple users to discuss issues with maintenance, other problems that may occur, or even simply plan for holiday events. Lasty, the rent feature of the app allows the landlord to view all paid rent bills as well as view which tenants are overdue payments.

All of the information that is being stored and retrieved is done so using a Firebase database. Firebase allows for all of the information to be stored online so that no local storage is being used up on the phone. In addition, Firebase enables the data the be constantly updated as soon as it detects changes to any of the values being shown. This is how the real-time messaging feature of the application is achieved as it updates the list of messages immediately after being notified that there is a new data entry. The images of signatures for the lease renewals are stored in the database as well. In addition to having real-time updates for all of the messages and different requests, there are also notifications provided by the app using a service that can be viewed in the notifications shade of Android. This alerts the user right away when a message or request is sent.

The interfaces are made using basic widgets and in some cases such as tenant layout, the interface is split into two LinearLayouts that each hold certain forms and the ListView respectively. In these cases, there is two separate layouts for portrait and landscape so the LinearLayouts can be vertical or horizontal depending on orientation. Through intents we are able to pass through the username, name, and an identifier in order to decide which layout would be used, tenant or landlord. These intents were also used in order to start new activities based on button clicks. Landlord layouts are simpler being comprised of a ListView and any other widgets that give the landlord specific capabilities. The notification system is built upon a service that will send a notification when data changes inside the database. The data received in most of the activities is done through a valueEventListener, and most of the ListViews are made through the FirebaseListAdapter. These give us enough control over the logic to make implementation easier. All of the buttons were implemented either using simple onClick functions or by creating an onClickListener. Some of the ListView items had onLongClickListeners attached to them so that more functions could be added to each of those lists.

Work Organization:

At the beginning of the project the work was more or less organized based upon either doing the landlord or tenant layouts respectively. As we progressed we noticed that many of the features could be reused between the two and later we decided that having us individually work on certain activities would be the best way to work on the project. At the end of the project, we all just chipped in randomly on certain pieces that needed to be accomplished as well as the merging.

We were able to use many concepts from class. Since we needed a database, we chose Firebase which we covered in remote databases. A lot of UI design was required, from the visual aspect to having defined landscape/portrait layouts as well as utilizing various widgets and getting them to all function correctly.. Obviously multiple activities were used, which was covered. We also used intents and bundles, as well shared preferences in multiple activities to accomplish different tasks within the app.

Appendix:

- Bill.java: Kristen Wong
- BillListAdapter.java: Kristen Wong
- DataSelectorFragment.java: Kristen Wong
- Landlord.java: Kristen Wong
- LandlordRentActivity.java: Kristen Wong
- Login.java: Brett Lechner
- MainActivity.java: Brett Lechner

- Maintenance.java: Brett Lechner
- NotificationService.java: Brett Lechner
- OnGetDataListener.java: Kristen Wong
- SignatureActivity.java: Tejas Bhoir
- Tenant.java: Kristen Wong
- TenantRentActivity.java: Kristen Wong
- TenantUtilitiesActivity.java: Kristen Wong
- WebActivityView.java: Tejas Bhoir
- ContactActivity.java:Brett Lechner
- CreateAccount.java: Brett Lechner
- LeaseActivity.java: Brett Lechner
- MaintenanceActivity.java: Brett Lechner
- MessagesActivity.java: Brett Lechner
- UtilityDateSelectorFragment.java: Kristen Wong

Firebase: https://landlord-tenant.firebaseio.com/
Landlord login: username = BL, password = b
Tenant login: username = tenant, password = t