6.1. Introduction

To promote friendship and proper communication between roommates, we introduce Hausmate. One-in-three U.S. adults have an adult roommate, with this number growing. Hausmate is a software which offers solutions to the problem of miscommunication. Computers offer unique capabilities for making communication easier. It is true that confronting problems in person is beneficial, and this application is no substitute for in-person interactions; rather, it is a platform for general organization and communication that aids relational management.

6.1.1 System Objectives

The objective of this application is to ease communication between roommates. Through shared lists and calendars, it allows roommates to keep track of the plans of other roommates and see what needs to be done as a house. This, along with announcements and messages, are displayed as a preview on the main screen for ease of access to the information.

6.1.2 Hardware, Software, and Human Interfaces

6.1.2.1 React Native

The mobile application framework used to develop Hausmate allows code sharing across iOS and Android OS, which aids in the development of both at the same time using knowledge of Javascript.

6.1.2.2 Firebase Service

Firebase is a service Hausmate uses to handle our database and backend. It allows us to update the database with new data depending on the input of the user, such as creating or updating lists or chats. It also allows us to retrieve information from the database easily.

6.1.2.2.1 Cloud Firestore

Cloud Firstore is a document-oriented database that stores the data for Hausmate. This data includes user data, haus data, lists, and chats. Cloud Firestore allows us to update and store data in real time.

6.1.2.2.2 Firebase User Authentication

Firebase User Authentication allows Hausmate to create and validate users through email and password. This service aids in the login of current users and the creation of accounts for new users.

6.1.2.3 Google Calendar API

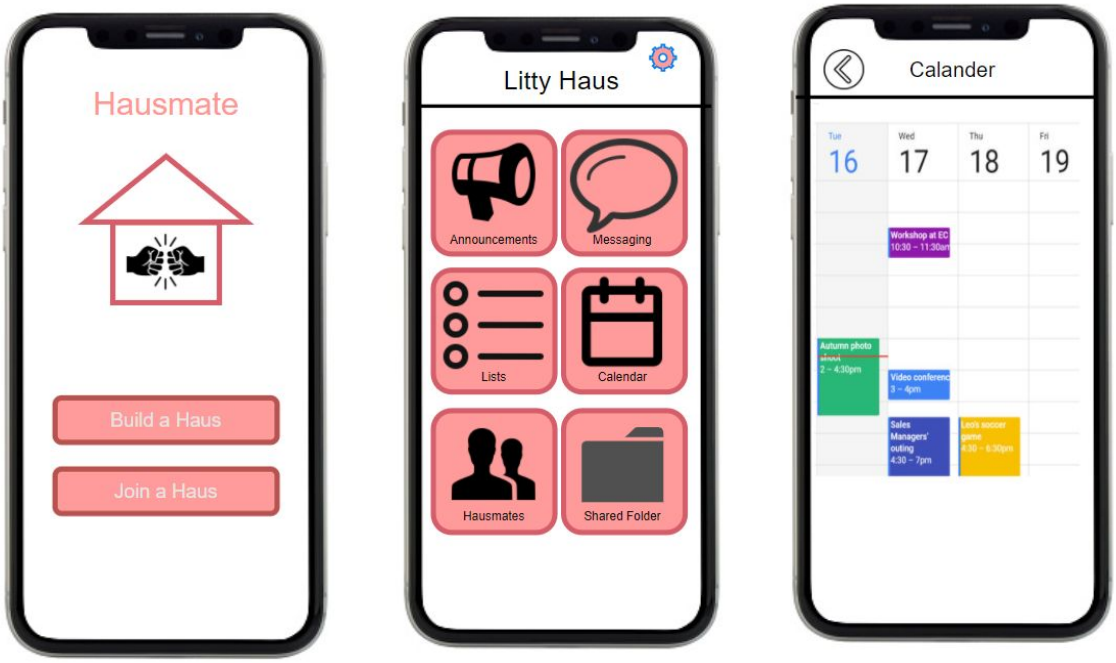
Hausmate uses the Google Calendar API to create its shared calendar tool. It allows users to edit a calendar that can be seen by the other members of their Haus. Users are able to add, remove, and view events that correspond to a day and a time.

6.1.2.4 Mobile Device

The hardware Hausmate is intended to run on are mobile devices, such as cellular phones. These Mobile Devices allow for various types of user interfaces, such as Touch and Text Entry.

6.1.2.7 GUI

The main Graphical User Interface of Hausmate is centered around the Main Screen. The Main Screen displays a preview of information that comes from all other tools, such as user statuses, recent messages, or upcoming events. This Main Screen displays tools, like the chat, lists, or shared calendar tools.



6.1.2.8 Touch

Using a Mobile Device, the user will be able to interface with the GUI using the touch feature associated with mobile devices. Users will be able to navigate through the Main Screen and utilize Tools by touching the associated buttons.

6.1.2.9 Text Entry

Using a Mobile Device, the user will be able to interface with text-fields in the GUI through a pop-up keyboard or connected keyboard to type their desired input.

6.1.2.10 Android OS and iOS

Through the use of React Native, Hausmate is designed to run natively on Android OS and iOS.

6.2 Architectural Design

The architecture of Hausmate can be divided into three subsystems: database control, user-to-user interaction control, and user-to-application control. While the dataset for our application does not require direct interaction with the system administrators, that is, our group, the latter two subsystems will ultimately depend on the maintenance of a server-based database hosted via firebase.

6.2.1 Major Software Components

6.2.1.1 Database Control

To delineate, database control refers to the classes and interfaces which access our group’s dataset, a distinction that encompasses the functional facets of Hausmate. All user input, messages, profile information, event items, and such, will be stored and will remain accessible so long as our database server stays active.

6.2.1.2 User-to-user Interaction Control

User-to-user interaction control is tantamount to the implementation of an SMS, which is based on a user’s values, personal information and such, for the corresponding attributes in the entities of our dataset. For example, direct messages between two individuals in a four-person Haus will be stored in correspondence within the area of the dataset containing those two users, which will be contained within the location of that four-person Haus.

6.2.1.3 User-to-application Interaction Control

User-to-application interaction control then refers to the manner in which an individual can asynchronously interact with other users. These interactions concern indirect forms of broad communication, such as creating announcements, setting events via Google Calendar, and posting on the anonymous board. The differences identified between this subsystem and user-to-user interaction may appear arbitrary, since all input is stored in similar fashion. However, the distinction is intended to benefit our group regarding the management of our database.

6.2.2 Major Software Interactions

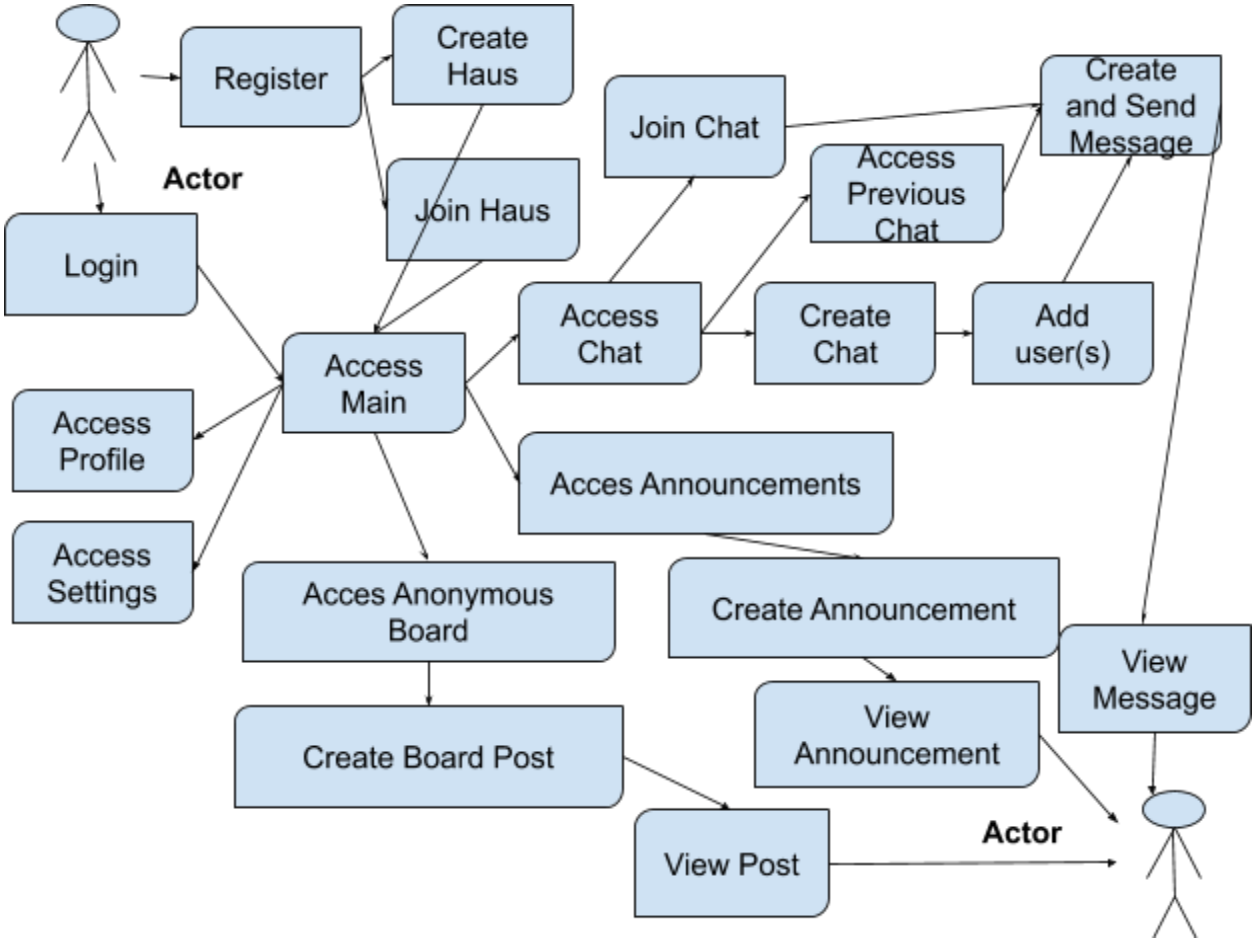
The files for our classes relate to an individual screen that appears in Hausmate, several of which are class files, that largely feature extensions of React components, references to the React API that are required in communicating with our dataset. The Register and Login Screens, two initial screens, pertain to their classes that contain methods for sending and retrieving information stored from an unaltered or altered area of our cloud-hosted database. A verification code is sent via an authentication system which allows a first-time user to create a profile, then they can join or create Haus, a way to further specify the attributes of a given user within our dataset. Each of the aforementioned screens consequently present event handlers that, again, allow the user to alter the information of our application’s dataset.

Once a user has established or verified themselves in the app, they are taken to the main screen, which displays a number of interactables. The chat feature serves as an in-house messaging client for Hausmate, where messages are stored in arrays that are once more sent to our database. Announcements, lists, and the anonymous board are other methods of communication among users, where asynchronously-created posts can be shared and viewed as they are placed in cloud-storage. Events are handled via an implementation of the Google Calendar API.

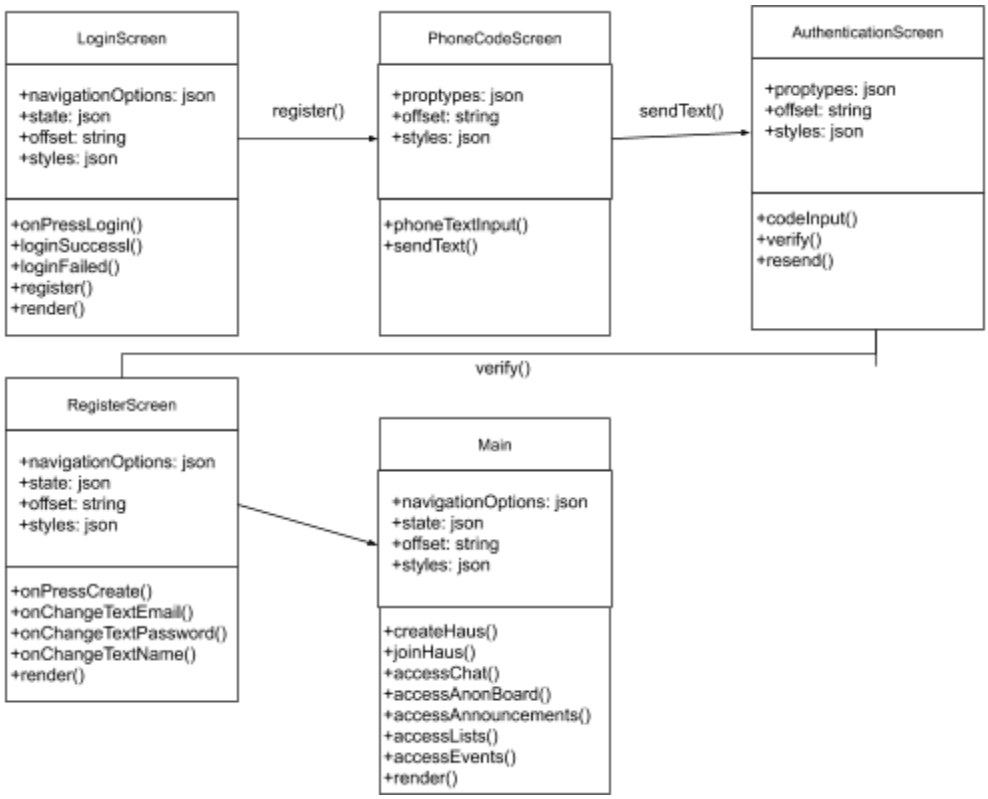
6.2.3 Architectural Design Diagrams

Below are the associated diagrams with Hausmate.

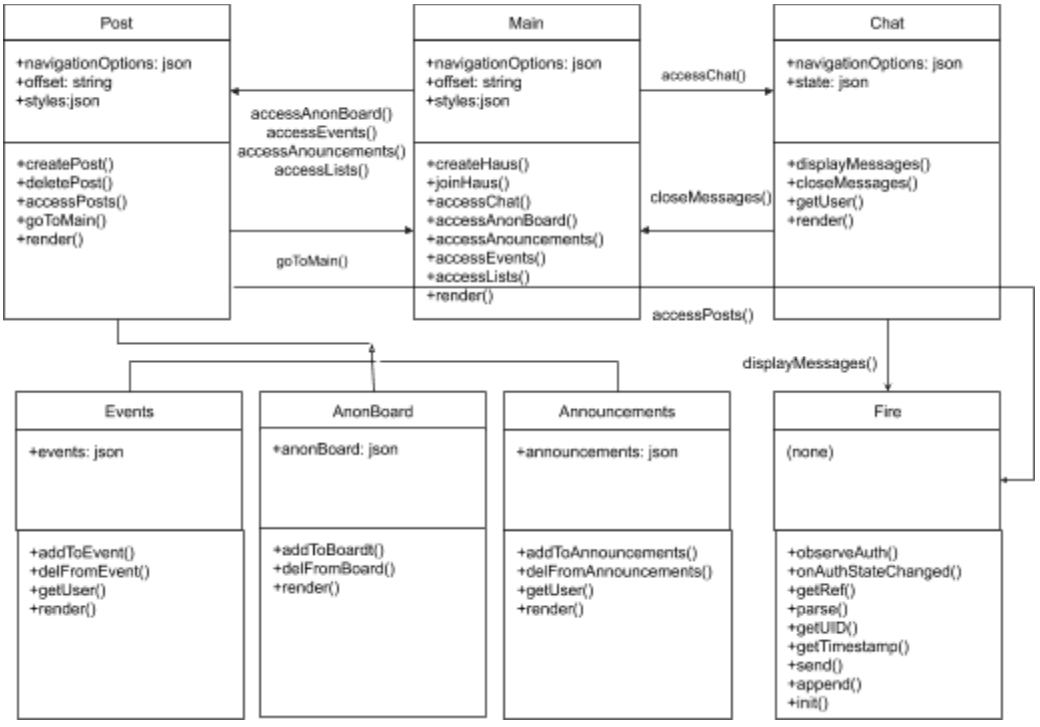
6.2.3.1 Use Case Diagram


Actor
Register
Create Haus
Join Haus
Login
Access Main
Access Profile
Access Settings
Access Chat
Join Chat
Access Previous Chat
Create and Send Message
Create Chat
Add user(s)
Acces Announcements
Acces Anonymous Board
Create Announcement
View Announcement
View Message
Create Board Post
View Post
Actor

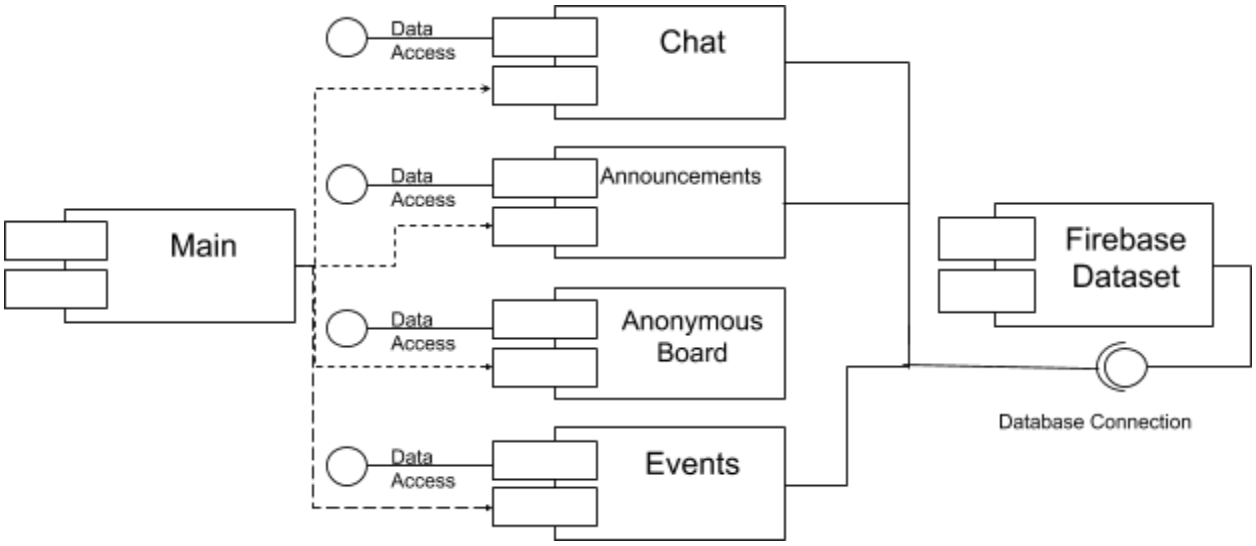
6.2.3.2 Class Diagram Part One



6.2.3.3 Class Diagram Part Two



6.2.3.4 Component Diagram



6.3. CSC and CSU Descriptions

The Hausmate project consists of three computer software components with a varied number of computer software units per CSC. Included in our CSCI are the Graphic User Interface, Server, and Database CSCs.

The GUI includes a number of modules, classes, and tools for its implementation, ultimately allowing an individual to view and interact with other users through direct messages, postings that will remain within the interface for a given time, and different forms of communication. Several CSUs follow the Haus Creation and Haus Joining modules, the corresponding modules for the Settings, Events, Lists, Announcements, Chat, and Status components of the application, and the Tools interface, which contains a series of classes and modules pertaining to various user interactions specific to the aforementioned components.

The Server CSUs concern several systems that both serve and define a user of the application. The notification system and integration of the Google Calendars API store and track data regarding synchronous and asynchronous events, respectively, while the Authentication and Database Systems serve to establish and

contain user and Haus entities. Each system pertains to a class of the same name.

The CSUs for the Database CSC feature entities for Hauses, Users, Lists, and List Entries, each featuring the appropriate attributes and keys.

6.3.1 Class Descriptions

The following sections provide the details of all classes used in the Hausmate application. Regarding the GUI, many are extensions of React components that utilize Expo and React APIs, and these each consequently feature the render prop function for the purpose of handling user interaction. Many classes also make use of our dataset, hosted via firebase, so data can be added, removed, and queried by the user.

6.3.1.1 RegisterScreen

Below are the listed fields and methods for the class RegisterScreen.

6.3.1.1.1 Fields

6.3.1.1.1.1 navigationOptions

This field is a static json object with keys describing different choices for either a first-time or returning user.

6.3.1.1.1.2 state

This field is a json object with keys describing the personal info that the user will input to create their corresponding Hausmate profile, including “name,” “email,” “password,” and “avatar.”

6.3.1.1.2 Methods

6.3.1.1.2.1 onPressCreate

This method is an asynchronous function that will alter the values of the state field in the case of valid inputs by the user.

6.3.1.1.2.2 onChangeTextEmail

This function alters the value for the “email” key in state to the string in the corresponding text field.

6.3.1.1.2.3 onChangeTextPassword

This function alters the value for the “password” key in state to the string in the corresponding text field.

6.3.1.1.2.4 onChangeTextName

This function alters the value for the “name” key in state to the string in the corresponding text field.

6.3.1.1.2.5 render

This function generates the styling, text fields, and buttons with which the user will view and interact, including fields for the user to enter the requested information and a button to confirm their submission.

6.3.1.2 LoginScreen

Below are the listed fields and methods for the class LoginScreen.

6.3.1.2.1 Fields

6.3.1.2.1.1 state

This field is a json object with keys describing the personal info that the user will input to create their corresponding Hausmate profile, including “email” and “password.”

6.3.1.2.2 Methods

6.3.1.2.2.1 onPressLogin

This method is an asynchronous function that invokes the .login function, returning a promise if the input user data corresponds to an existing user.

6.3.1.2.2.2 loginSuccess

This method invokes the .navigate function, which takes the user to their respective main menu screen, if the promise from onPressLogin is successfully returned.

6.3.1.2.2.3 loginFailed

This method alerts the user that the input information is invalid; that is, a user object does not exist in the dataset which equals the input.

6.3.1.2.2.4 render

This function generates the styling, text fields, and buttons with which the user will view and interact, including fields for the user to enter the requested information and a button to submit.

6.3.1.3 ChatScreen

Below are the listed fields and methods for the class ChatScreen.

6.3.1.3.1 Fields

6.3.1.3.1.1 navigationOptions

This field is a static json object with keys describing the title of the chat group and its users.

6.3.1.3.1.2 state

This field is a json object with a key of an array containing all recorded messages.

6.3.1.3.2 Methods

6.3.1.3.2.1 displayMessages

This method connects to the dataset in firebase and displays recorded messages among the users from previous sessions.

6.3.1.3.2.2 closeMessages

This method disconnects from the firebase dataset.

6.3.1.3.2.3 getUser

This method parses the name and user id of the user for the use of a React API (GiftedChat in this case).

6.3.1.3.2.4 render

This function generates the styling and text fields with which the user will view and interact, including a field for the user to enter messages.

6.3.1.4 MainScreen

Below are the listed fields and methods for the class MainScreen.

6.3.1.4.1 Fields

6.3.1.4.1.1 state

This field is a json object which contains all pertinent information to a Haus in the database

6.3.1.4.2 Methods

6.3.1.4.2.1 render

This function generates the styling and widgets the user will view and interact with from the Main Menu.

6.3.1.5 SettingsMenu

Below are the listed fields and methods for the class SettingsMenu.

6.3.1.5.1 Fields

6.3.1.5.1.1 state

This field is a json object which describes all settings properties and their current values

6.3.1.5.2 Methods

6.3.1.5.2.1 updateSettings

This method replaces current values for properties with any upcoming changes.

6.3.1.5.2.2

This method generates the styling and interactable components to view and alter from the Settings Menu

6.3.1.6 ProfileScreen

Below are the listed fields and methods for the class ProfileScreen.

6.3.1.6.1 Fields

6.3.1.6.1.1 state

This field is a json object which describes all profile properties and their current values.

6.3.1.6.2 Methods

6.3.1.7 ToolsMenu

Below are the listed fields and methods for the class ToolsMenu.

6.3.1.7.1 Fields

6.3.1.7.2 Methods

6.3.2 Detailed Interface Descriptions

6.3.2.1 LoginScreen

The LoginScreen is first opened when the user opens the application. The user interfaces with the login screen via text-fields to input their login information. After the user is authenticated, the user is navigated to the MainScreen.

6.3.2.2 RegisterScreen

The RegisterScreen is accessed through the LoginScreen via a button. The user is able to interface with the RegisterScreen through text-fields, where the user can input their information to register their account. When confirmed, a user will be created and added as a user to Cloud Firestore database.

6.3.2.3 CreateJoinScreen

The CreateJoinScreen is accessed after a user is registered through the RegisterScreen. The current user is passed to the CreateJoinScreen. After choosing to create or join a Haus, the current users UID will be added to a Haus. The user will then be navigated to the MainScreen.

6.3.2.4 MainScreen

The MainScreen is accessed through the LoginScreen or the CreateJoinScreen. The current user and their associated Haus is passed to the MainScreen. The user is able to navigate to other Tools through the ToolsMenu, edit their profile through the ProfileScreen, or edit application settings through the SettingsMenu.

6.3.2.5 ChatScreen

The ChatScreen is accessed through the ToolsMenu. When opened, data on the current user and their Haus is passed. The user will be able to create or continue chats with members in their Haus. After inputting their desired message via text-field, their message is stored via Cloud Firestore.

6.3.2.6 SettingsMenu

The SettingsMenu is accessed through a button on MainScreen. The user will be able to change settings of the application, such as notifications. These updated settings will be reflected in the Cloud Firestore database. The user will be able to log out via the SettingsMenu which will navigate the user to the LoginScreen.

6.3.2.7 ProfileScreen

The ProfileScreen is accessed through a button on MainScreen. The current user data will be passed. The user will be able to change their profile information or status. Any changes will be reflected in the Cloud Firestore database.

6.3.2.8 ToolsMenu

The ToolsMenu is accessed through a button or Touch swipe on the MainScreen. The ToolsMenu will display icons of tools available to the user, such as the Calendar, List, or Messaging Tools. When navigated to, data on the current user is passed.

6.3.3 Detailed Data Structure Descriptions

6.3.4 Detailed Design Diagrams

6.4 Database Design and Description

The database will contain information on users and Hauses. Each User entity will have an associated UserID (UID), email, and status. Each Haus entity will contain the list of users by UID, shared lists

associated with the haus by ListID, ID of the shared calendar, announcements, and active chats.

6.4.1 Database Design ER Diagram

6.4.2 Database Access

The database will be accessed via methods of the Firebase Service. These methods allow us to change, update, and remove information on the database depending on the action of the user. If a user or Haus is created, a new user or haus entity will be added to the database. If any member of a Haus changes a shared list or calendar, that list under the ID of the current Haus will be updated to reflect the changes.

6.4.3 Database Security

Hausmate plans to use encryption for a user’s email and password to secure a user’s personal information. Since Google’s Firebase Service is being used as the main hub for our database and backend, we are relying on their security to protect our information.