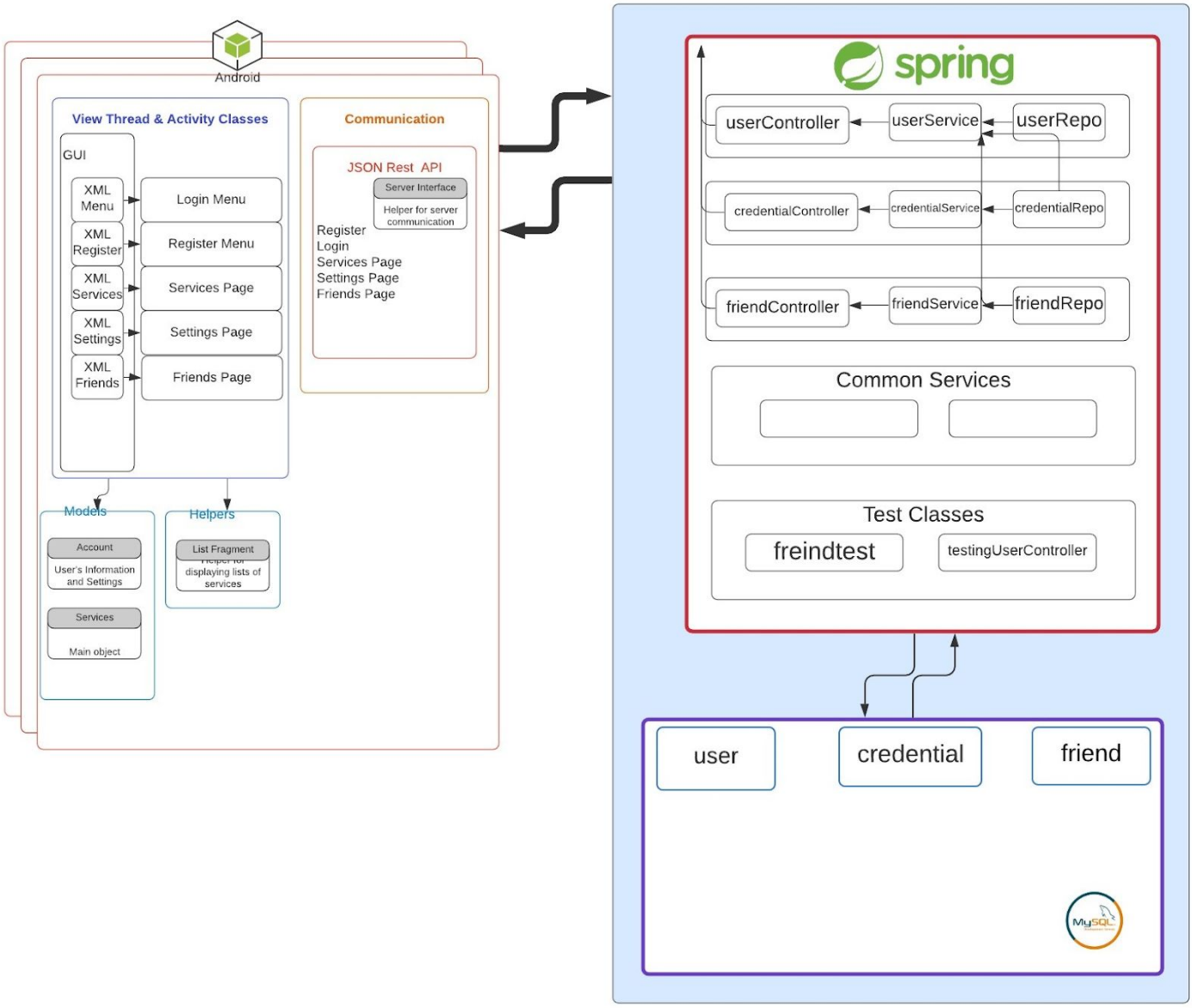
Block Diagram

HV_7 - Jon Schnell, Isabel Maymir, Brent Brain, Ben Kelly

Bunker


Android
View Thread & Activity Classes
GUI
XML Menu
XML Register
XML Services
XML Settings
XML Friends
Login Menu
Register Menu
Services Page
Settings Page
Friends Page
Communication
JSON Rest API
Server Interface
Helper for server communication
Register
Login
Services Page
Settings Page
Friends Page
Models
Account
User's Information and Settings
Services
Main object
Helpers
List Fragment
displaying lists of services
spring
userController
userService
userRepo
credentialController
credentialService
credentialRepo
friendController
friendService
friendRepo
Common Services
Test Classes
freindtest
testingUserController
user
credential
friend
MySQL

Design Description:

Backend:

Models:

The three primary data structures that are present in Bunker are user, credential, and friend. The user model holds all user data including name and password as well as one to many mappings to friend and credential. The credential entity holds credential information. This allows users to add many credentials. Similarly the friend entity allows users to have many friends. This will ultimately allow a user to share a credential with a friend. And retrieve owned credentials at any time from anywhere.

Controllers:

Each entity has a controller associated with it. The purpose of these controllers is to set up the parameters of each API we implement to be used by the frontend application. These endpoints will allow the frontend application to accomplish necessary tasks such as retrieving owned credentials and log in. These endpoints are how two frontend and backend communicate over the internet.

Services:

Each entity also has a service associated with it. These classes provide a layer of abstraction between the repositories and the controllers which makes testing easier and makes the code more modular. The services are where the magic happens so to speak, for example instead of autowiring directly from the repository to the controller we put the service in between. This allows us to write algorithms that we can more easily call in the controller to accomplish a task.

Frontend:

Modules:

Registration:

The Module allows a user to register an account with the bunker application and be assigned a unique UUID such that when they login on subsequent occasions their information is always correct and preserved. This also provides security for the user.

Login:

This Module allows a user to enter previously defined information in order to be logged in to the system and receive customized information and services which are stored under their UUID.

Services:

This Module holds a comprehensive record of all passwords, usernames, and service names corresponding to each user. This information is ONLY accessible once the user is logged in and provides the correct UUID to the backend server.

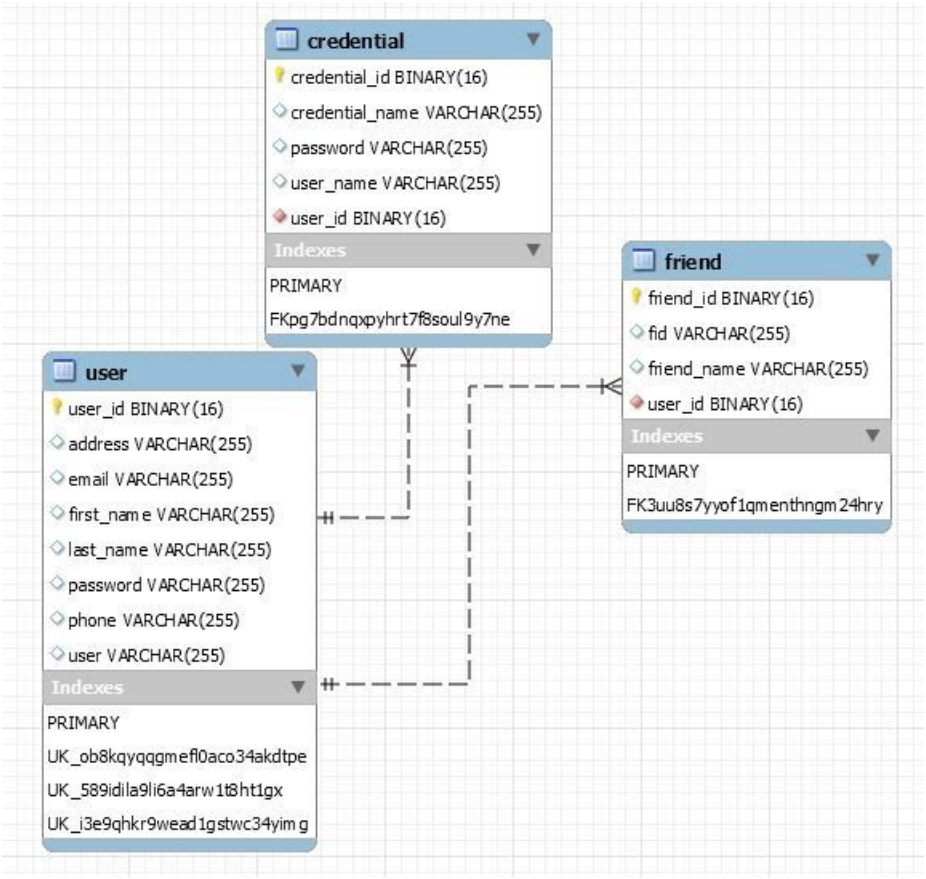
Settings:

This Module gives the user control over their account, allowing them to delete their account, and change important account information or application settings as needed.

Friends:

This Module allows for a user to view their current friends, add friends to their current friends, remove friends from their current account, and change which passwords/information is shared with each of their friends.

Tables and Fields:



User:

- **User_id - generated ID for user. Used as key.**
- **Address - address of the user**
- **Email - email of the user**
- **First_name - first name of the user**
- **Last_name - last name of the user**
- **Phone - phone number of the user**
- **User - username of the user**

Credential (one to many from user):

- **Credential_id - generated ID for the credential. Used as key.**
- **Credential_name - name of the credential**
- **Password - password associated with the credential**
- **User_name - username associated with the credential**
- **User_id - ID of the user the credential is associated with. Used to map back to User.**

Friend (one to many from user):

- **Friend_id - generated ID of the friend object**
- **FID - userID of the friend**
- **Friend_name - name of the friend**
- **User_id - ID of the user who has this friend. Used to map back to User.**