Design Document for <<Tenant Direct >>
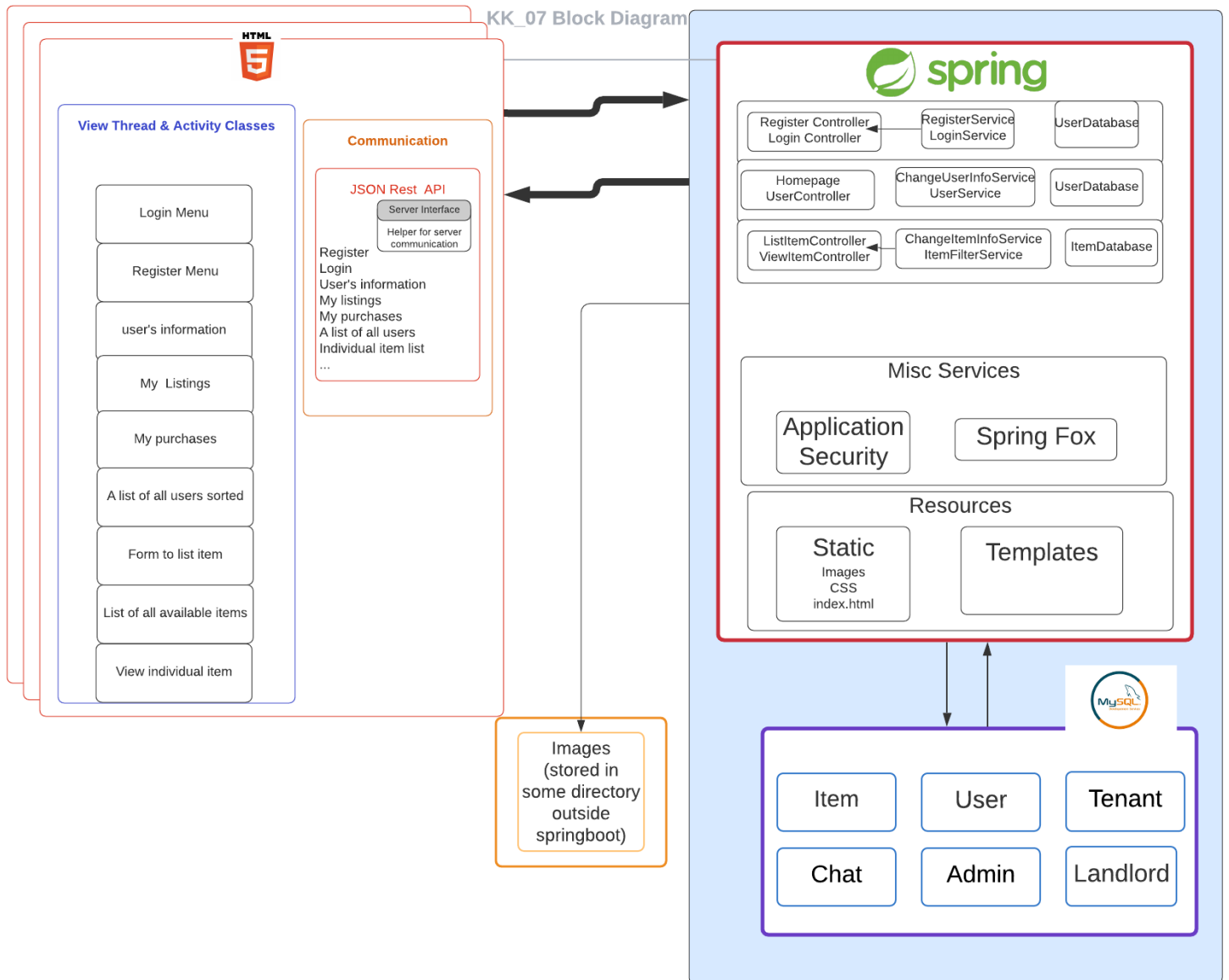
Group <KK-07>

Member1 Name:Jeet Nair

Member2 Name:Samuel Mason

Member3 Name:Zihao Huang

Member4 Name:Jue Wang

View Thread & Activity Classes

Login Menu

Register Menu

user's information

My Listings

My purchases

A list of all users sorted

Form to list item

List of all available items

View individual item

Communication

JSON Rest API

Server Interface

Helper for server communication

Register
Login
User's information
My listings
My purchases
A list of all users
Individual item list
...

Images
(stored in some directory outside springboot)

spring

Register Controller
Login Controller

RegisterService
LoginService

UserDatabase

Homepage
UserController

ChangeUserInfoService
UserService

UserDatabase

ListItemController
ViewItemController

ChangeItemInfoService
ItemFilterService

ItemDatabase

Misc Services

Application
Security

Spring Fox

Resources

Static

Images
CSS
index.html

Templates

MySQL

Item

User

Tenant

Chat

Admin

Landlord

On the client side are using HTML with Thymeleaf to parse it. The spring boot application serves the webpages by serving up the view that it rendered in the browser. The user interacts with these web pages by using form to post information back to the server, clicking on links and buttons to navigate around the layout. Multiple clients can access the site at the same time. Each client will connect to the server through the Tomcat server through the HTTP rest controllers.

The user signs up by filling out the form that is served up when they navigate to the register section of the sign in page. This information is transferred to the server via the RegisterController which receives the post request mapping. RegisterController then calls the RegisterService method that takes the information (name, email, phone number, password, ect) and using the UserDatabase interface puts the information into the User table of the SQL database.

The user can then login with the username and password that they chose previously. When this information is posted the LoginController will receive the post request and will call a method in LoginService that will check the email and password that was entered in the login page match an existing email and password that is already in the database. If there is a match the user will be redirected to their home page.

Once in the home page the user can view all their information, their listings, their purchases, and can view a list of all users. The user can click on List and Item and will be able to post an item in the same fashion as the Register Process. The items will go in an Item table in the SQL database. The user can upload images for their item that they are listing. The file will be assigned a number (likely the id of item in the Item table) and will be written to a directory on the server (not in the folder in the spring boot project where static images are kept) and this file path will be saved.

When the user clicks on "see available items" from their homepage, they will see a list of available items. The server will simply give Thymeleaf the list of Java objects and the Thymeleaf parser will be capable of enumerating these. The user can click on any item in this list to view that item. They will be redirected to view this item and can see any pictures displayed alongside the item information. They also can view the name of the seller that listed this item and can click a link to chat with the user. The chat will be implemented using websockets.

The view users functionality comes with sorting. The user can choose to view other users based on Email, Address, Phone, Building, ect. The sorting function is in the Userservice. The sorting takes place and the list of User objects is given to Thymeleaf to enumerate and display.

item
- 🔑 id INT
- ◇ description VARCHAR(255)
- ◇ filepath VARCHAR(255)
- ◇ length_in_days VARCHAR(255)
- ◇ name VARCHAR(255)
- ◇ price VARCHAR(255)
- ◇ sellerid VARCHAR(255)
- Indexes ▶

tenant
- 🔑 id INT
- ◇ apartment VARCHAR(255)
- ◇ email VARCHAR(255)
- ◇ f_name VARCHAR(255)
- ◇ l_name VARCHAR(255)
- ◇ password VARCHAR(255)
- ◇ phone VARCHAR(255)
- ◇ unit VARCHAR(255)
- Indexes ▶

admin
- 🔑 id INT
- ◇ apartment VARCHAR(255)
- ◇ email VARCHAR(255)
- ◇ f_name VARCHAR(255)
- ◇ l_name VARCHAR(255)
- ◇ password VARCHAR(255)
- ◇ phone VARCHAR(255)
- ◇ unit VARCHAR(255)
- Indexes ▶

land_lord
- 🔑 id INT
- ◇ apartment VARCHAR(255)
- ◇ email VARCHAR(255)
- ◇ f_name VARCHAR(255)
- ◇ l_name VARCHAR(255)
- ◇ password VARCHAR(255)
- ◇ phone VARCHAR(255)
- ◇ unit VARCHAR(255)
- Indexes ▶

user
- 🔑 id INT
- ◇ email VARCHAR(255)
- ◇ password VARCHAR(255)
- ◇ type VARCHAR(255)
- Indexes ▶