**Product Design**

|  |  |
| --- | --- |
| **Team** | f610-01c **Team members**: Ibrahim Mujhid,  Pooja Apparao Raghoji,  Raghuram Gopalkrishnan,  Rahul Kumar shinde,  Rohan Kerkar. |
|  |  |
|  |  |
|  |  |
|  |  |

# Architectural Overview

# 

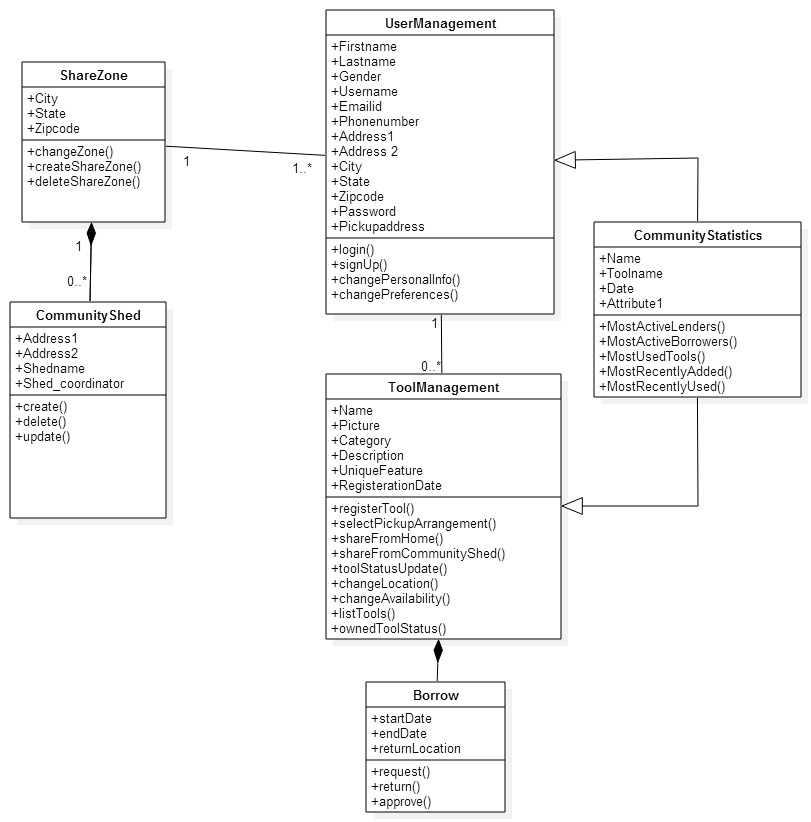
**Database Design**

# 

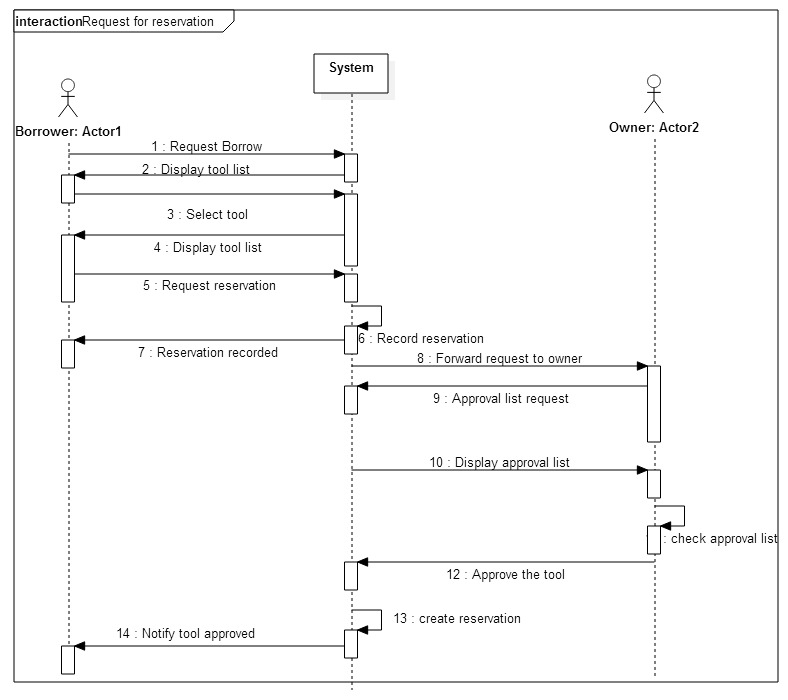
# Components and Functions

|  |  |
| --- | --- |
| User | 1. The component is responsible for authenticating registered users. 2. The subsystem allows successfully logged in users to have access to the tool share system.   Classes and modules Used:   1. BootstrapAuthentication 2. Profile 3. Preference 4. signUp |
| Tool | This component handles the following:-   1. Tool Registration 2. Tool Management 3. Tool Listing 4. Tool Availability   Classes and modules Used:   1. RegisterTool 2. ToolListing 3. ToolManagement 4. ToolAvailability |
| Shed | This component handles:-   1. Shed creation 2. Shed update 3. Shed listing   Classes and modules Used:   1. Shed creation 2. Shed Management 3. Shed listing |
| Statistic | This component handles:-   1. Most Active Lender Statistics 2. Most Active Borrowers. 3. Most used tools. 4. Most recently used tools 5. Most recently added tools.   Classes and modules used.   1. Activestatistics 2. Usedstatistics 3. Recentstatistics |

# Class Diagram(s)



# Sequence Diagram(s)



# Design Rationale

1. **Tool for Uml Diagrams:** We had a prolonged discussion of the tool we would use for constructing the Uml Diagrams. We narrowed our choices down to ‘Star UML’, ‘Visual paradigm’, ‘Argo UML’. We then selected ‘Star UML’ to construct UML diagrams for ease of using tools and editing our diagrams in Star Uml.
2. **Database:** We decided on using the database ‘sqlite’ which is inbuilt in python. The reason for this decision was that we would be able to develop the application faster by using the inbuilt database.The database table names and attributes were initially confusing and redundant. We have tried to reduce redundancy by normalizing the tables that we created initially.
3. **Class Diagrams:** To follow the divide and conquer strategy, we made multiple classes such as toolListing, changeLocation, sharefromhome, sharefromcommunityshed. But we realized that these can be functions in the Tool Management class that we have created. So we included them under one class, as the User will use all these functions under Tool management in the project.
4. **Names of Classes: Naming Convention:** The names of the classes while constructing the class diagrams initially were not specific to the classes that we needed to implement. We changed the names of classes to specific names so that we get a better understanding of what our classes do. Similarly we changed some of our data members and member function names to specific names for better understanding.
5. **Sequence Diagram:** The sequence diagram of ‘Request for reservation of tool’ was initially an overview of how objects interact in time sequence. We discussed the use cases related to the reservation of tool in detail and noted the actions we need to consider from the systems point of you as well as the user owner and borrowers point of view. We then modified our sequence diagram to show a better control flow with respect to what the actors and lifeline would do.
6. **Django + Python:**- The python and django framework was initially difficult to understand with respect to including the models and views in one file or to separate them to resolve merge issues. We decided to have them separate.
7. **Pillow:**- Instead of using the built in django image functions we decided to use pillow because of the ease of implementation.
8. **Built in authentication:** We initially decided to use custom authentication for the handling of user registration and authentication, however as we progressed we decided to use the in built authentication system as it reduces the overhead involved in coding the authentication system.
9. **Compartmentalizing views and models:** We decided to compartmentalize the views and models into separate files based on the functionalities of the components. This was done with the view of separating the functionalities thereby enabling multiple users to work on different parts of the functionality without having to worry about merge issues.
10. **Separating of databases:** We decided to separate share zone from the user and shed so as to prevent the data redundancy.