Online Auction

System Design Document

V0.7

02.12.2018

Berkay YILMAZ

Tolga GÜLDÜTUNA

Feyzullah Berkay DANIŞ

Tekin EVRİM

Prepared for

SE301 Software Engineering



**1. Introduction**

Online Auction is a auction application that is developed for android platform. It designed to become collection of components that communicate ﬂexibly with each other since there is no complex tasks in the application to do.It designed with React using Javascript to write to code and translate the software to an android output. The application itself is a simple ccmposition of familiar websites of ebay, gittigidiyor , letgo etc. Since the auction applications is not used too much in our contry we also get information to develop our application by looking universal applications or websites that is doing the same task. We wanted to use layered architecture on our project also our design goals are as follows;

The system should be easy to use by under the means of usability also it must support multiple users to supply needed performance to users. OA also need to be reliable since it has to do with auctions. The app must be secure to be under the terms of reliability.It also needs to be supportable by many versions of the android the reach more users.

**1.1. Purpose of the System**

The main purpose of the applicaion is to provide registered users a platform that they can easily and fastly their stuff. OA also a good opportunity to bring the auctioning style shoppining in to the local area and make people’s items worth as much as they do to prevent them from being scammed on or prevent from stuck with unselling the item.

**1.2. Design Goals**

***Usability***

Since the end-user will be using the system while anytime in day even in work , it is essential for the system to be intuitive and easy to use.

***Multiple Users***

The application should support tasks that are performed by multiple users in concert, supplying each with the necessary information at the appropriate time.

***Reliability***

OA should be provide all of its users a secure experience. Unregistered visitor should be able to see the current auctions and stuff. Yet, users need to be registered user to do bidding and selling procession. Logging to the application should be provided with unique e-mails and password that are appropriate for password criteria. The membership information should be private and should not be shared with anyone without the will of the user.

***Performance***

OA should be responsive in different versions of Android OS and it should be able to scale correctly in different versions of Android OS and devices. It should be running in more than one mobile device . OA is going to be dynamic content, so there should not be complicated queries in back end to not decrease performance.

***Understandability***

OA must be easy to understand for users to all ages.Bidding or Selling functions must be most 2-3 clicks events.

***Supportability***

OA should be managed by admin. The application should be supported on different android versions and be independent by hardware mostly. In the react native part system should be able to open for new implementations easily.

***Implementation***

There are no constraints on the hardware platform. There are no constraints imposed by the maintenance team. There are no constraints imposed by the testing team. The design methodology is obtained as Agile Development Approach. System runs with query. JavaScript language is used in this system. Query based statements are handled by React Native. JavaScript, ………..

***Legal***

The Online Auction does not use any license or licensed work. It is a student project.

**1.3. Definitions, Acronyms, and Abbreviations**

*SDD :* System Design Document

*OA :* Online Auction

*React :* **React Native** is a JavaScript framework for writing real, natively rendering mobile applications for iOS and Android

*Admin:* System actor that administrates the system.

*Visitor:* System actor that is not a registered user of the system.

*Subsystem:* Collection of classes, associations, operations and events closely related to each other.

*HTTP :* The HyperText Transfer Protocol

*UI :* User Interface

**1.4. References**

[www.gittigidiyor.com](http://www.gittigidiyor.com)

[www.ebay.com](http://www.ebay.com)

www.tr.letgo.com/tr

[www.liveauctineers.com](http://www.liveauctineers.com)

**2. Current Software Architecture**

There are some websites and applications that are related to our application . These applications are to provide auction bidding and selling platform between registered users. The applications and websites mentioned are;



The common applications are big and focusing on a bigger area. But our application is only focusing on auctioning in our local area. Since there is no application that is serving only this service in our local area this is a good opportunity for our application. The common problems also in letgo there will be here too. We need to prevent users to be scammed in an auction process. Since we are not responsible for any payment and we are just a platform that helps registered users to sell their stuff with auctioning , the money transections will be in the users responsibility. Besides from that local people are not used to use auctioning to be a part of their lives. But this could initially turn into a opportunity.

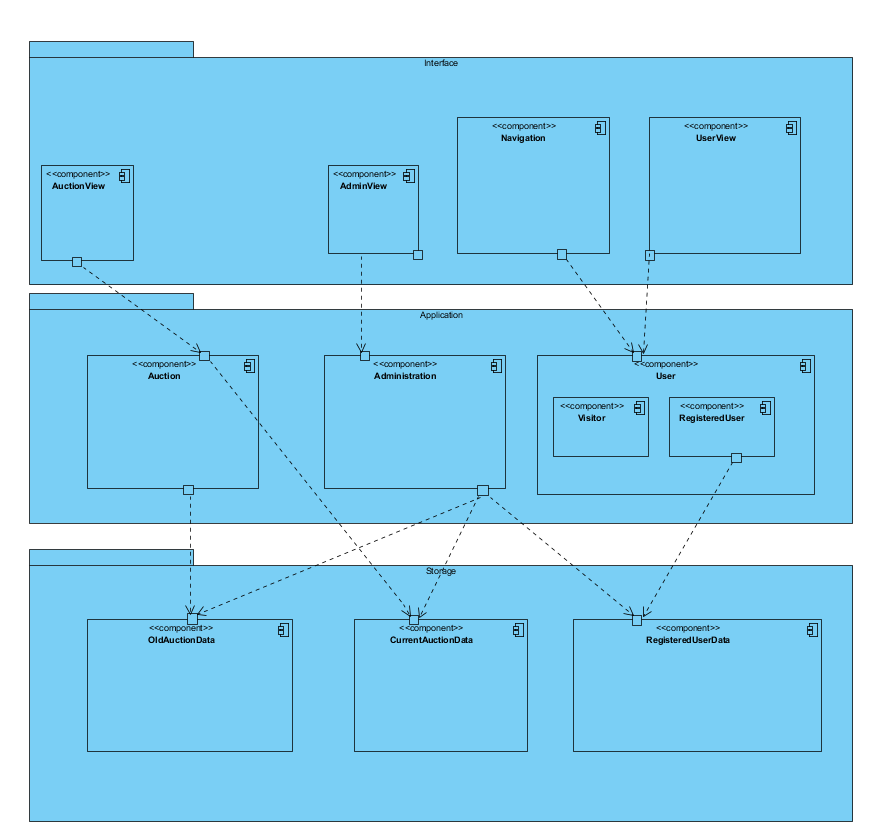
**3. Proposed Software Architecture**

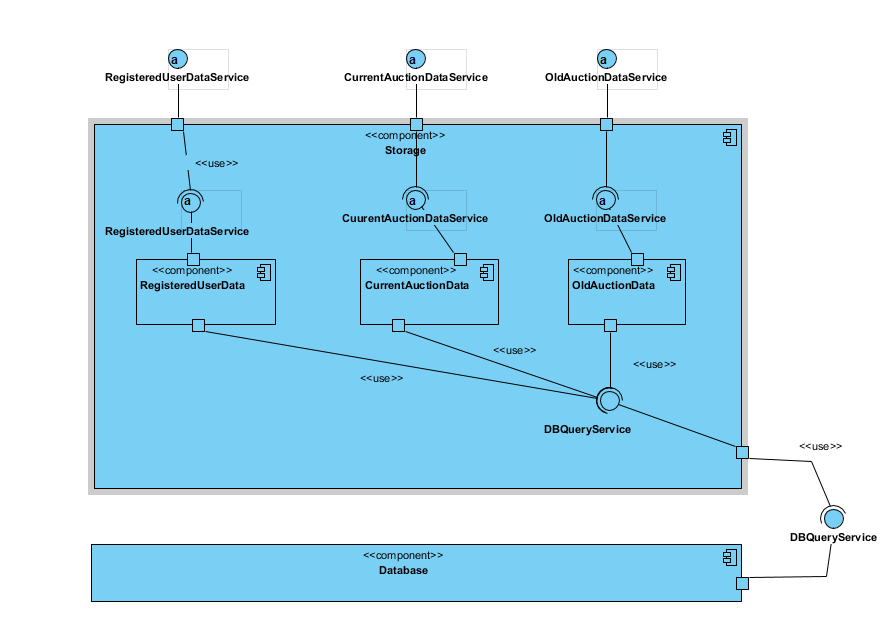
Documents the system design model of the new system.

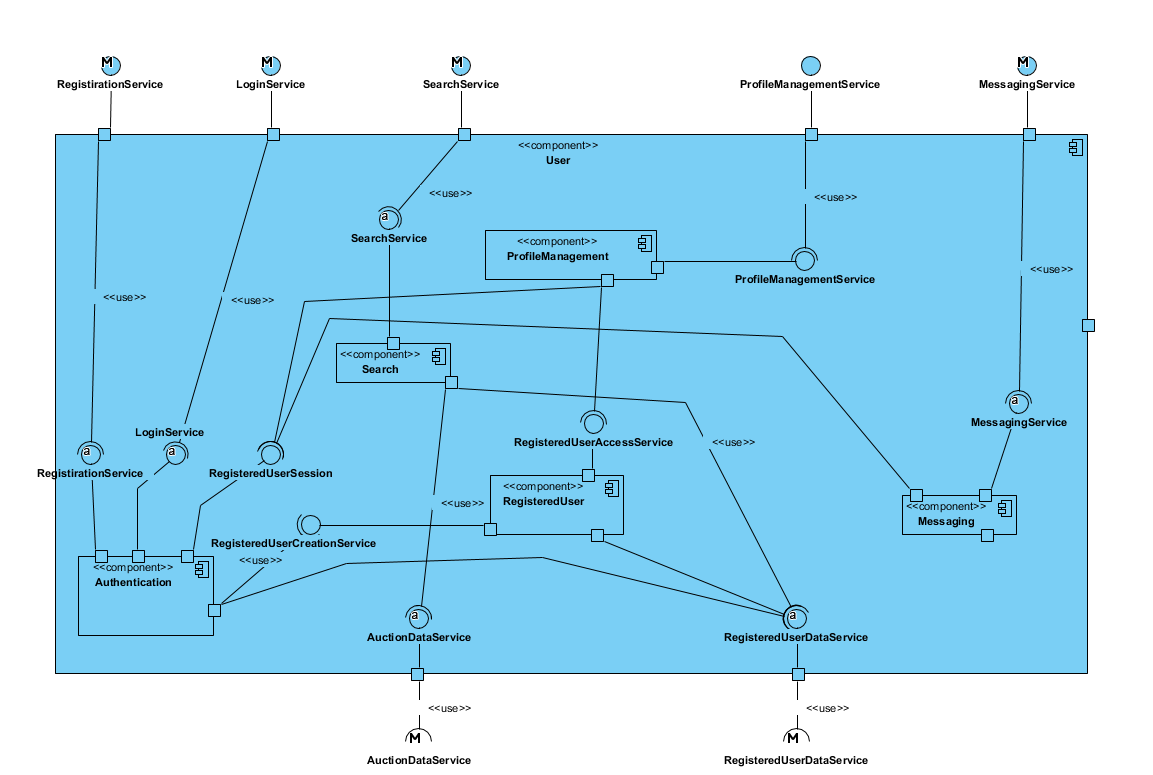
**3.1. Overview**

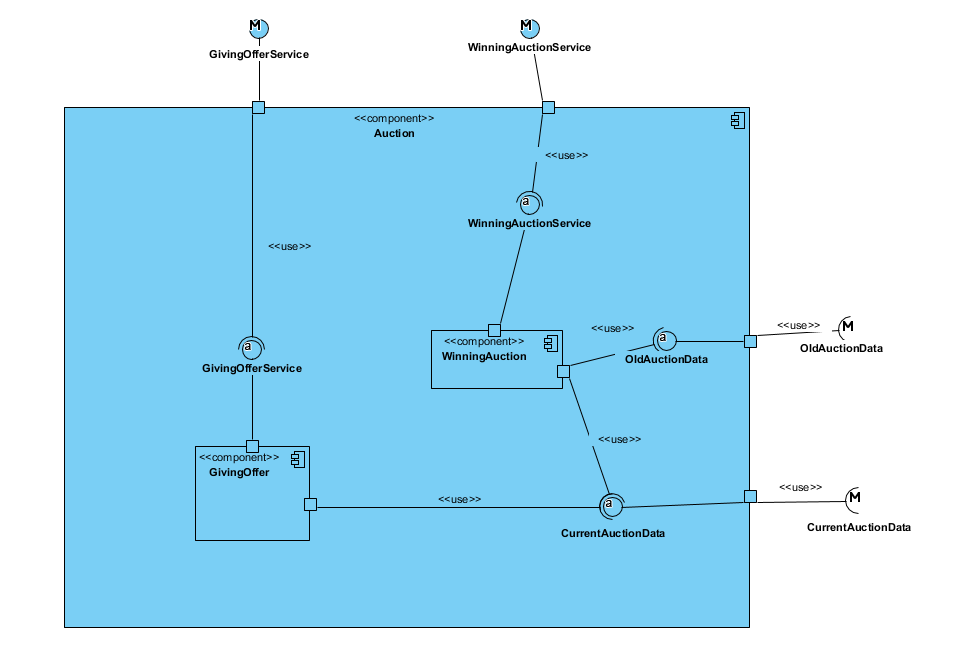
Present a bird’s-eye view of the software architecture and briefly describes the assignment of functionality to each subsystem.

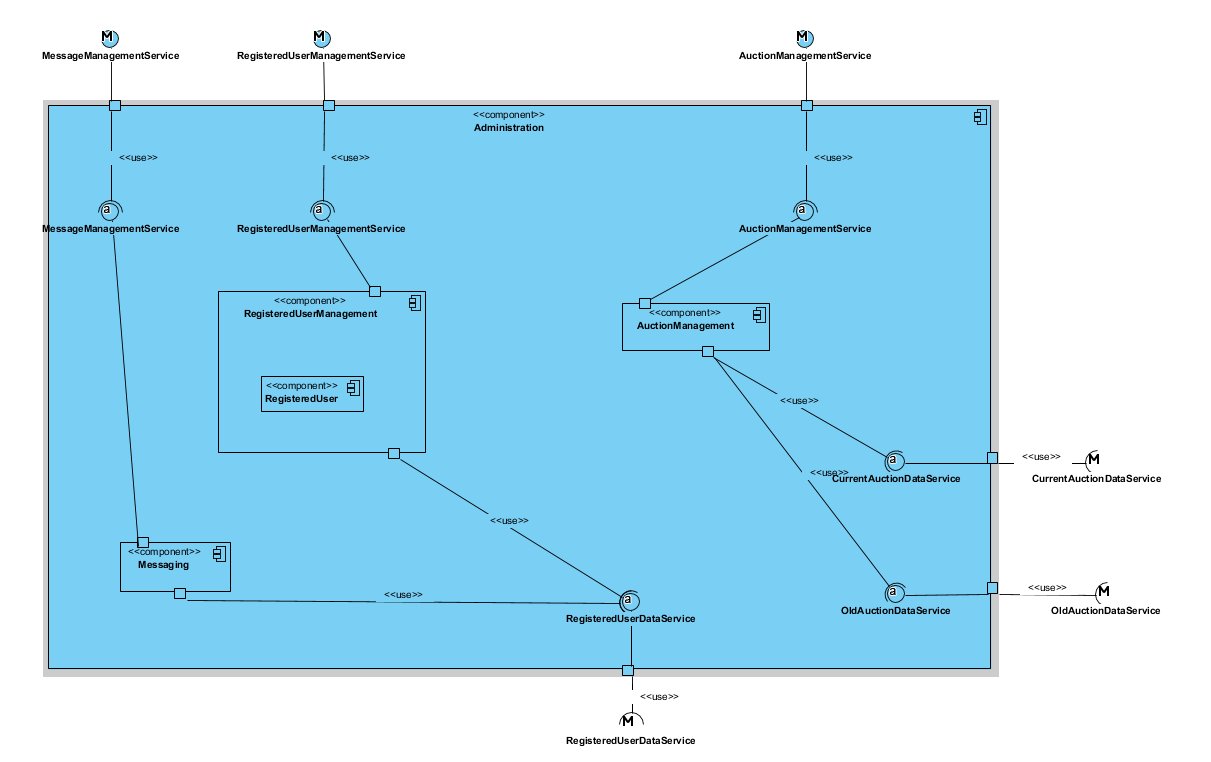
**3.2. System Decomposition**

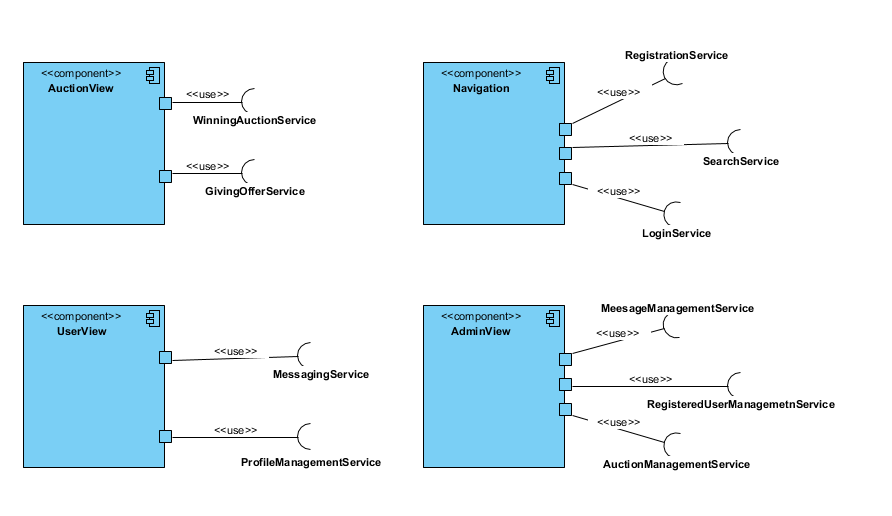


****

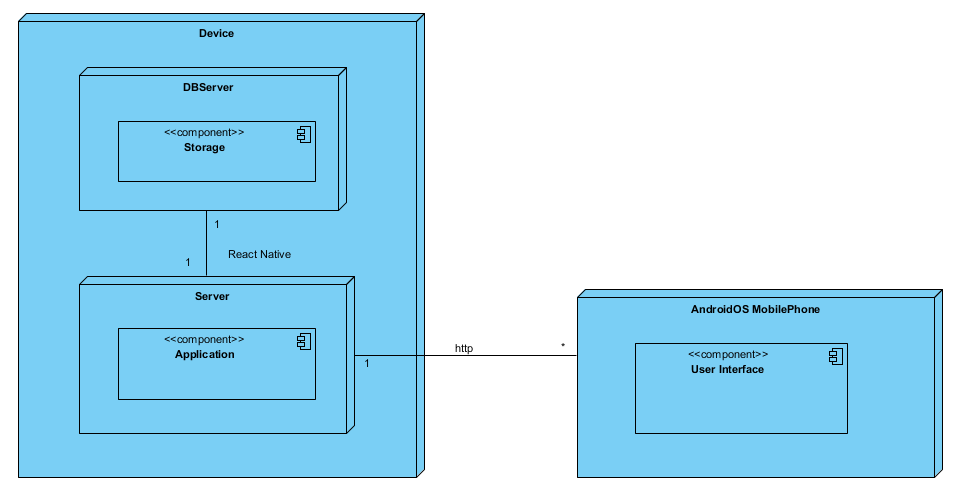
****

****

****

****

**3.3. Hardware Software Mapping**

****

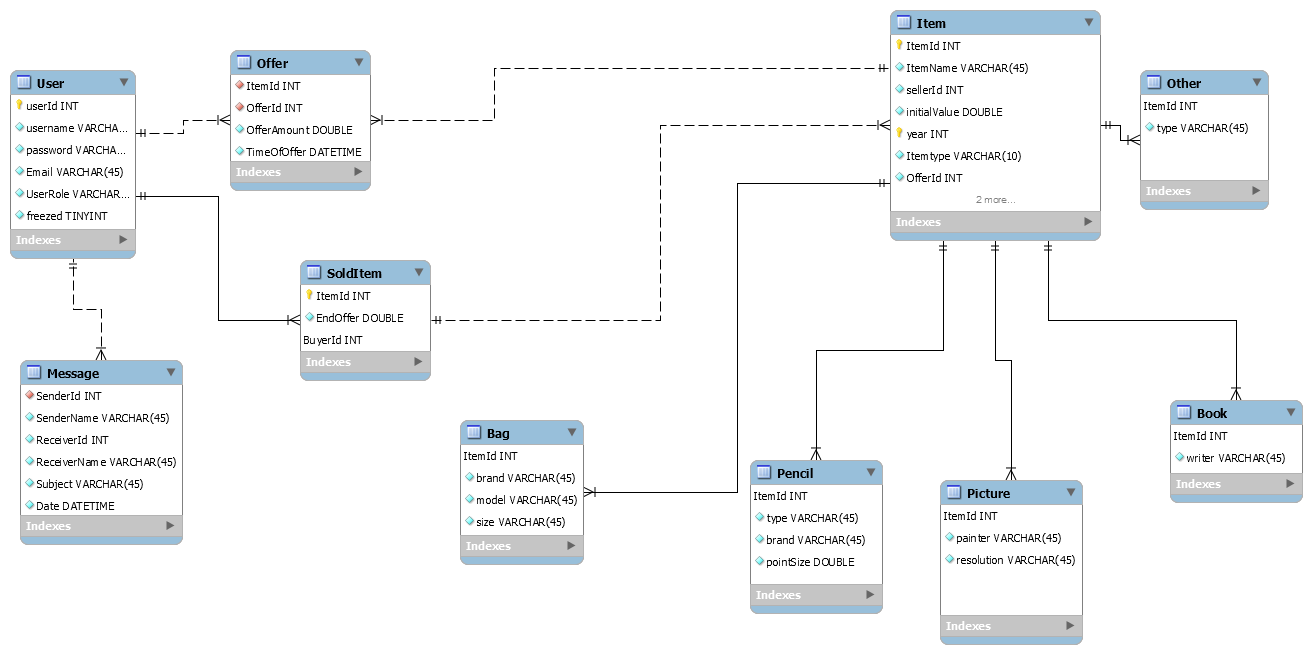
Describe how subsystems are assigned to hardware and off-the-shelf components. It also lists the issues introduced by multiple nodes and software reuse.

**3.4. Persistent Data Management**

The system needs persistent storage of its users, their information, offers, items and their relations with each other And these data must be accessed repeatedly - many times by multiple users - and each request should be returned as fast as possible. For these reasons, using a Relational Database Management System is a must.

While the initial userbase may be small, the system storage infrastructure should be scalable for a growing userbase. Data security is also another concern.

For these reasons, we will be using MySQL as our Database Management solution which covers all of our needs and more. Also, it’s Community Edition - which we use, is open source under the GNU General Public License, helping us reduce costs. It’ll also create an access queue for the data it holds, which allows for reliability in concurrent access scenarios.



The data scheme for the database is mainly made up of its users, their offers and the auction created by registered user’s. The ‘user’ table holds information needed from each and every user, password (hashed) etc. The ‘registered user’ tables hold additional for each user type’s needed information. This might be an registered user “About Us” text or a auction’s offers. The Admin user resides in neither of these two tables and is the sole entry of ‘user’ table.

‘Item table’ included subtable of the item types.Their are bag, pencil, book, bag and other items.

Finally, there is the ‘messages’ table that hold the messages received by each user .As for encapsulation, the Storage Layer ‘glued’ atop the Database will provide query-independent access to data for the higher layers.

**3.5. Access Control and Security**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Administration | User | Auction |
| Admin | ApproveAuction()  RejectAuction()  FreezeUser()  viewUserInformation()  viewUsersList()  viewNGOs()  messageUser()  viewMessages()  sendMessages() | login()  search()  viewItemInfo() | ViewAuctionInfo() |
| Registered User |  | login()  forgetPassword()  search()  UpdateProfileInformation()  viewItemInfo()  viewMessage()  sendMessages() | GiveOffer()  CreateAuction() |
| User (Visitor) |  | register()  search()  viewItemInfo() |  |

Users must provide an at least 6 characters long password to register. The database shall hold the password information hashed. As the system is not designed to hold sensitive information for users, no other encryption is required.

**3.6. Global Software Control**

Describe how the global software control is implemented. In particular, this section should describe how requests are initiated and how subsystems synchronize. This section should list and address synchronization and concurrency issues.

* Administration subsystem initiates OldAuctionData, CurrentAuctionData and RegisteredUserData subsystem to access/add/edit database data.
* User subsystem initiates RegisteredUserData subsystem to access/add/edit database data.
* Auction subsystem initiates CurrentAuctionData subsystem to access/add/edit database data.
* Navigation subsystem initiates Authentication subsystem for registration and login operations.
* Navigation subsystem initiates Search subsystem to search for running auctions.
* UserView subsystem initiates ProfileManagement subsystem to allow the user to interact with his/her profile information.
* UserView subsystem initiates the Messaging subsystem to allow the user to access his/her messages and/or message the Admin.
* AdministrationView subsystem initiates UserManagement, CampaignManagement or DonationManagement subsystems to manage each respectively.
* AdministrationView subsystem initiates UserManagement or CampaignManagement subsystem to approve user registrations or campaign creations respectively.
* AdministrationView subsystem initiates MessageManagement subsystem to view admin messages and allow the admin to message other users.
* CampaignView subsystem initiates CampaignEditor to create and edit NGO created campaigns.
* CampaignView subsystem initiates CampaignViewer for retrieving campaign information for viewing purposes.

The database queues each query which provide concurrency control for storage.

**3.7. Boundary Conditions**

Startup:

- First of all , admin is declared with react native.

Logging in:

- Username and/or password field are blank.

- Password is not 6 characters long.

- Password and username don’t match.

- Username is wrong or does not exist.

- Main Page does not appear after logging in.

Register:

- Username is already in use

- Password is not 6 character long

- Username and/or password and/or password again fields are blank.

- Password and Password again are not match

Profile Management:

- User can’t edit his/her information or the changes do not reflect.

- System crashes while editing profile information.

Starting Auction:

- Registered user send a request for a new auction without selecting a category.

- Registered user send request for a new auction without filling the empty blanks about the information of the item.

Giving Offer:

- Entered bid amount is lower than current bid.

- Bidding ends while making another bid.

- System crashing while bidding process.

**4. Subsystem Services**

Describe the services provided by each subsystem. Although this section is usually empty or incomplete in the first versions of the SDD, this section serves as a reference for teams for the boundaries between their subsystems. The interface of each subsystem is derived from this section and detailed in the Object Design Document.

**5. References**

The following is an example of listing a book in this section. Check the text to see how it is cross referenced (The whole document is based on [1]).