IITD : Complaint Managment System

Rajat Kumar Surana (2013TT10953)
Arpit Jain (2013TT10905)
Waghmare Saurabh Milind (2014CS10268)

March 10, 2016

To use this Complaint registration app, user should be registered in user database. This application can be used by 3 types of users.

- Admin(student).
- Student.
- Warden or Dean

1. If user is admin then he can use this app add new users to database. Also because admin is student he will have all student rights.
2. If user is student then he can use this for
   - File a new complaint.
   - View status of complaints(his old personal complaints or public complaints).
   - Vote for public complaints.
3. If user is warden or dean then user can use app to view complaint and to change status of complaint (Ex. warden can change status of hostel level compliant to **Resolved**, **Inprocess of resolving**, **Discarded**).

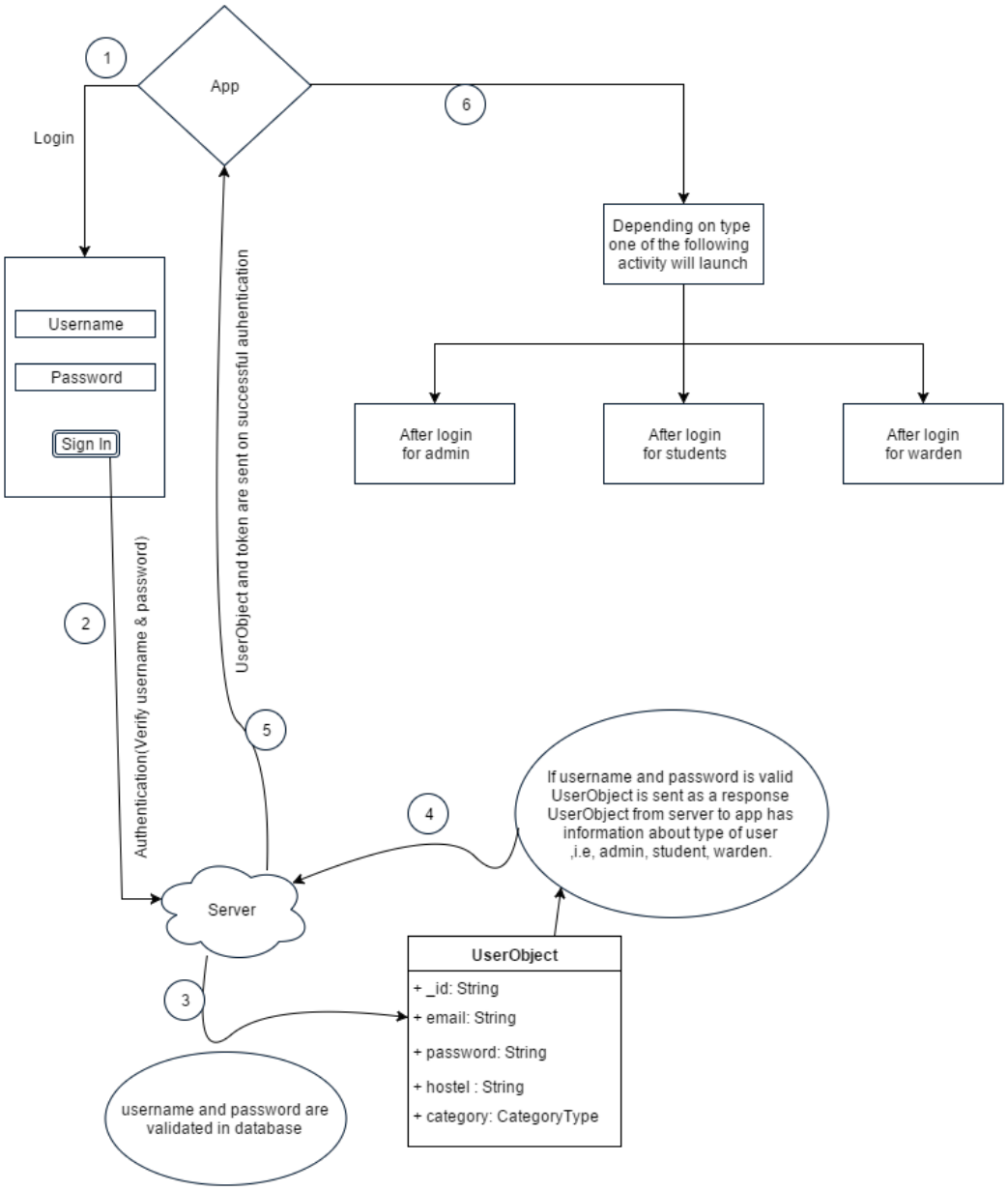


Figure 1: Overview of app

1 User Interface

- There is one Login screen in app for user to sign in. This screen consist of 2 Edittext for **Username** and **Password** . And a button for **Sign**

in.

- After user signed in depending upon type of user, he will see after login screens. There are 3 types of after login screens.

  1. After login for Student
  2. After login for Admin
  3. After login for Warden or Dean

- **After login for Stident** :-

  After login for student is a Navigation drawer activity. Navigation drawer has following items :-

  1. My complaints
  2. Notification
  3. Search complaint
  4. Subscribe
  5. Logout

  When user selects particular item from navigation drawer corresponding fragment will open.

  - **My complaints** :- This fragment shows list of old complaint of user. This list of complaints have information about Type of complaint, Description, Status of complaint.

    Thers is also one floating button at bottom right of this fragment(named "Add"). After clicking on this button a new fragment(File New Complaint) will open. Using this fragment user can post a new complaint.

    File new complaint has following layout :-

    * Drop down list "Type" of various possible type of complaints
    * Edittext "Place" for place (ex. Room no).
    * Editetext "Description" in which user decribea his complain.
    * User can also insert a image for complaint.
    * Button "OK" to post the complaint.

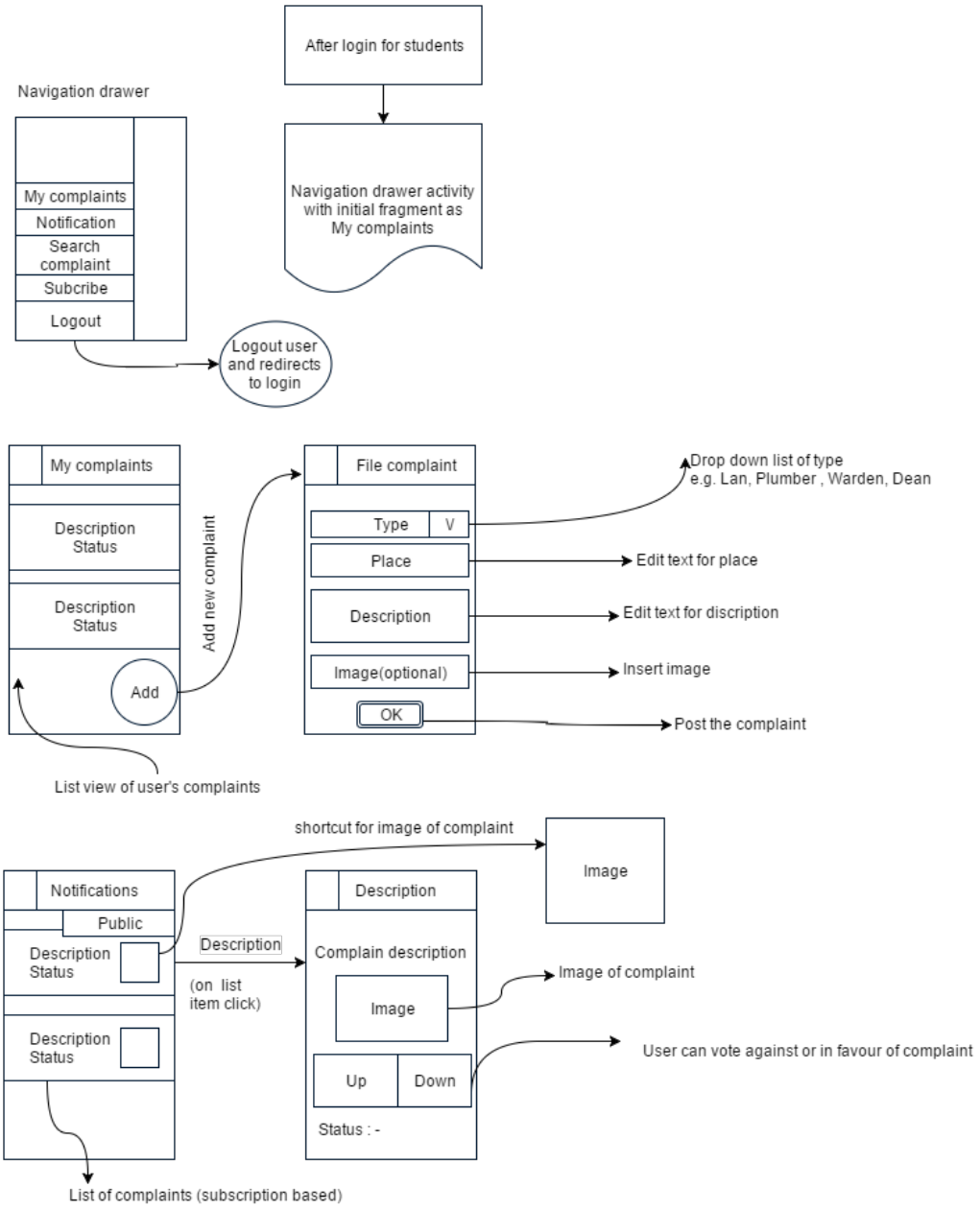


Figure 2: After login for students

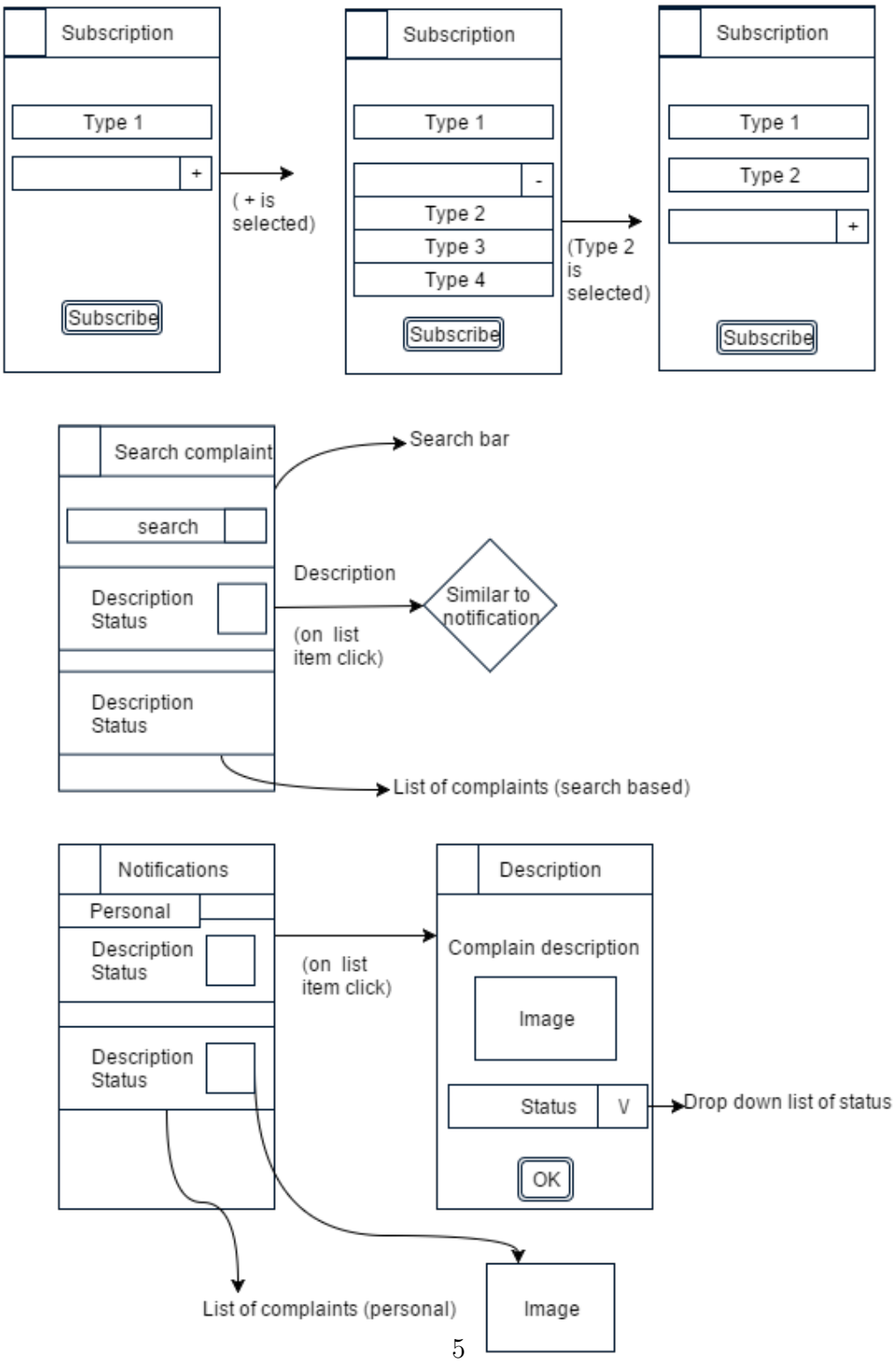


Figure 3: After login for students

- **Notification** :- This fragment contains 2 tabs. First tab have list of notifications of complain which are personal. And second tab have list of notifications of complains(of specific type to which user has subscribed) which are public.
  - Personal Notification :- Personal complaint list shows description of complaint, status and shortcut for image(if any). When user clicks on any of list item then a discription fragment opens which has complain description, image(if any), drop down list of status. So student can set status of complaint to unresolved or resolved.
  - Public Notofications :- Public complaint list shows description of complaint , status and shortcut for image(if any). When user clicks on any of such notification from list then new "Description" fragment will open. This fragment have complaint description, image(if any), voting buttons and status.
    There are two types of voting buttons **thumps up** and **thumps down**. When user selects particular voting buttons then app send this response to server.
- **Search complaint** :- Search complaints fragment have a search bar and list view simliar to public notification. Users gets a list which shows discripton, place and image(if any) shortcut. Only diffrence is type of complain is search based(not subscrintion based). On click of list item "Description" fragment will open same as in public complaint notifications.
- **Subscribe** :- This fragment have list of type subscriptions and insert bar below them. When user click on insert bar, a drop down list will be shown.After user login we get user's hostel and topic's drop down will be populated by sutaible suggestions. This list will have types which user haven't subscribed. When user selects particular types from drop down then this item will get added to our subscription list.
- **Logout** :- Logouts user and redirects to Log in screen.

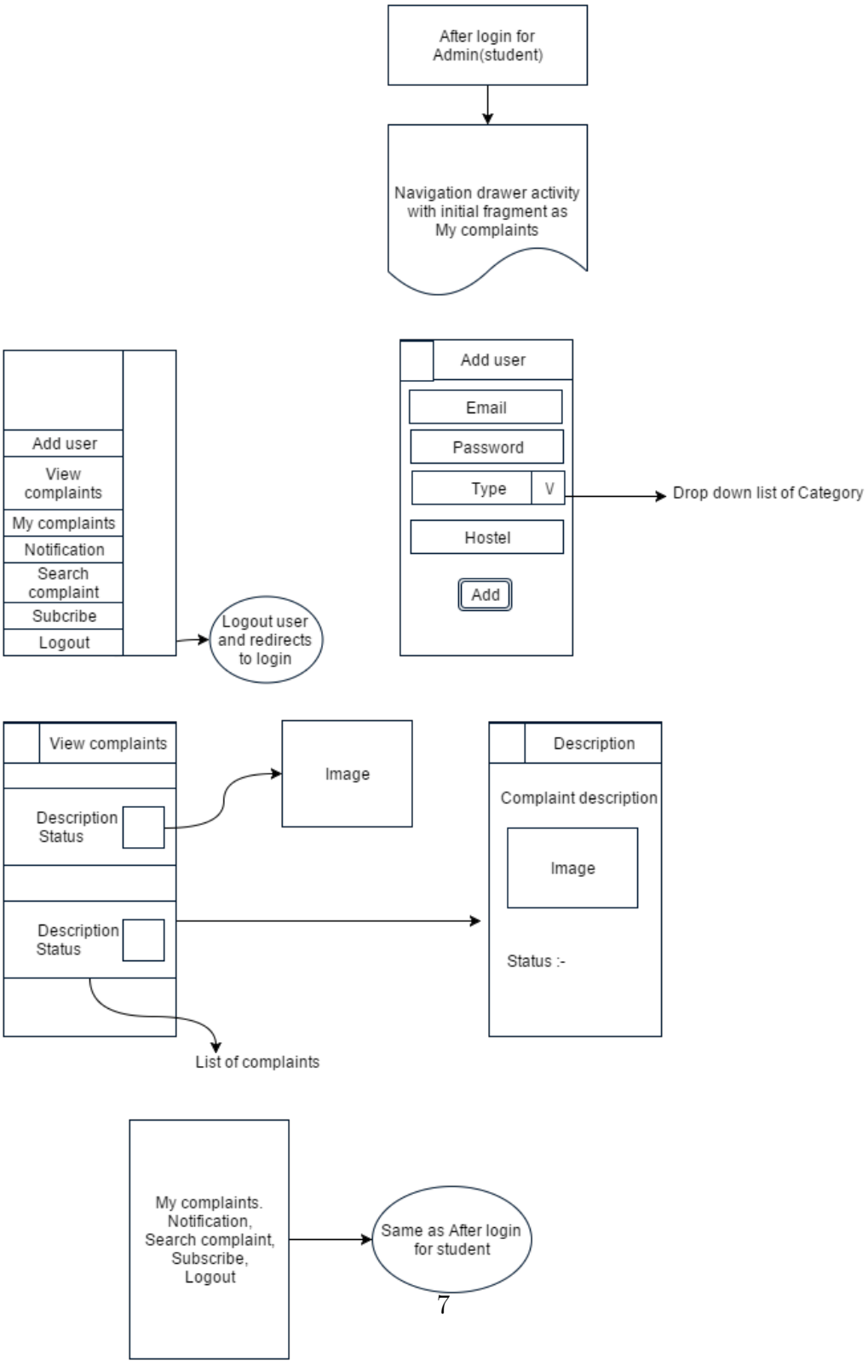


Figure 4: After login for Admin

- **After login for Admin** :-

  After login for admin is also an Navigation Drawer activity. In navigation drawer of acitivity there is one more extra item than items of navigation drawer of After login for student activity named **Add user**. When user selects this Add user item a fragment "Add user" will open.

  This Add user frgamnet will have following layout :-

  - Edittext email for email of user
  - Edittext password for password of user
  - Drop down list for type of user (type can be one of the Student, Admin(student), Warden, Dean).
  - Edittext hostel for hostel of user.
  - Button "OK" to add user to database.

  All others items of navigation drawer are same as items of navigation drawer of After login student activity.

- **After login for Warden or Dean** :-

  This also will be a navigation drawer activity. Navigation drawer will have following items :-

  1. View all complaints
  2. Search complaint
  3. Logout

  When user selects particular item from drawer corresponding fragment will open.

  - **View all complaints** :- Warden will get list of all complaints that are related to hostel to which he/she is assigned and Dean will get list of all complaitns that are Institude level. This list will show short description of complain, type of complain and shortcut for image(if any). When user selects particular complain from list then "Description" fragment will open(this fragment is slightly diffrent than "description" fragment of of student and admin).

    This fragment will have following layout :-

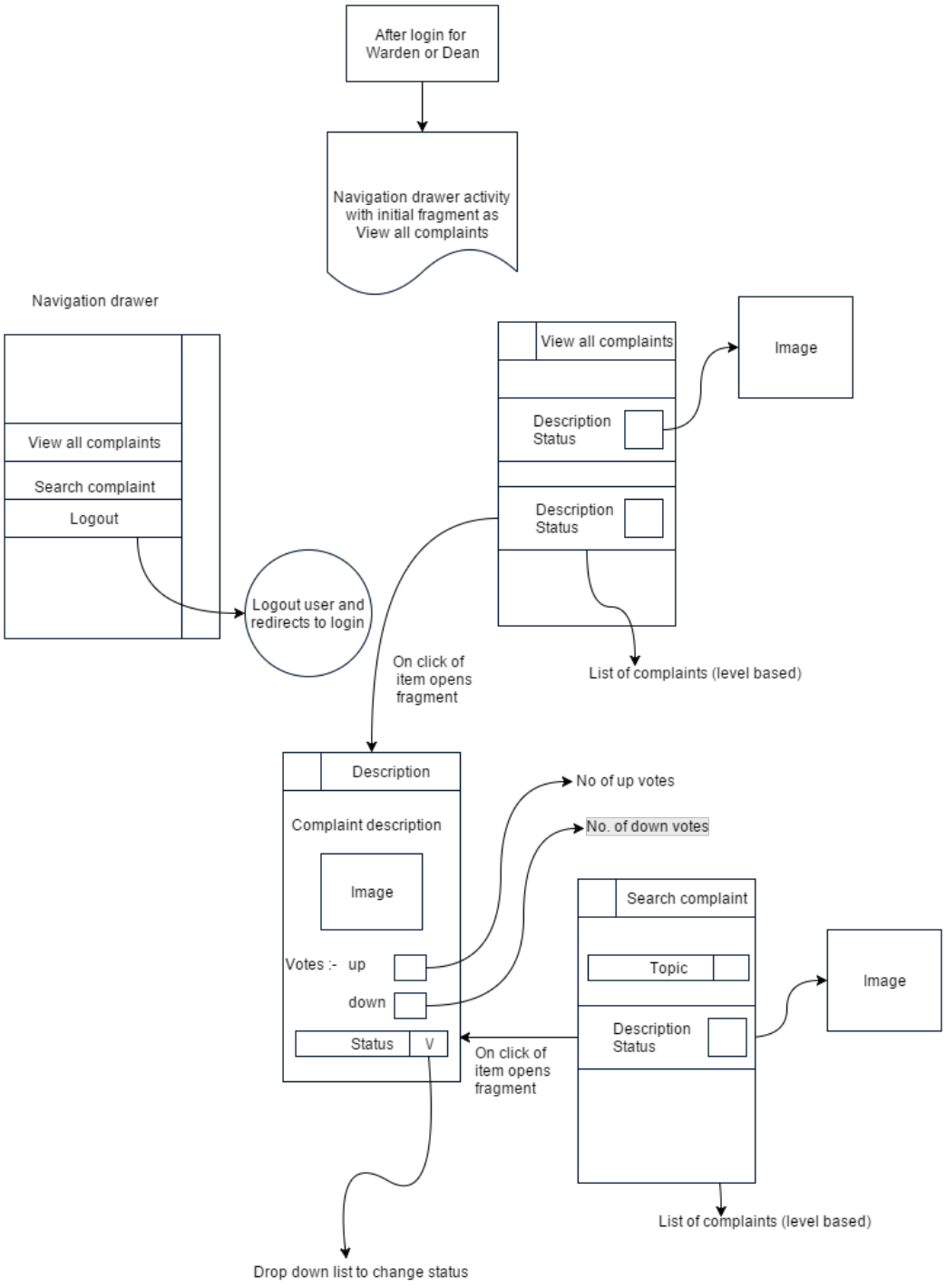


Figure 5: After login for Warden or Dean

    - Textview showing complain description.
    - Imageview showing image(if any).
    - Textview showing number of thumps up and number of thumps down votes
    - Drop down list to set status of complaint. User can set status as "Filed", "Discard", "In process" or "Resolved". If user set status other than "Filed" then voting for that partiuclar complaint will be closed.
  - **Search complaint** :- Search complaints fragment have a search bar and list view simliar to list of View all complaints. But item of list will be populated depending on type of search(ex . if warden search mess then all complaits that are related to mess of his/her hostel will be populated in list). On click of list item "Description" fragment will open same as Description fregament of View all complaints fragment.
  - **Logout** :- Logouts user and redirects to Log in screen.

2 Implementation Details

- **Organization of database** :- There only server database(as shown in fig.)
  - **Server database** :- Server database contains following objects :-
    1. UserObject
    2. ComplaintObject
    3. VoteObject
    4. userVote

    And following enumeration :-
    1. Category Type
    2. Status Type
    3. Solver Type
  - Due to enumerations in database we can detect invalid entries by user.

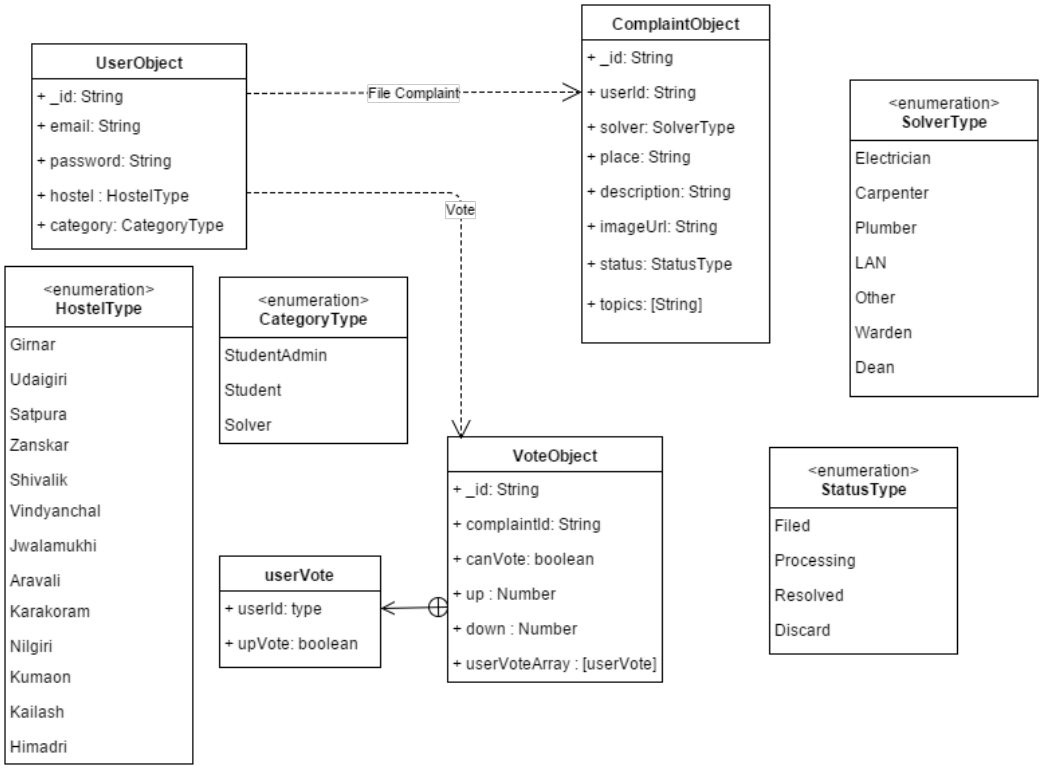


Figure 6: Database

- **Organization of APIs** :- All api except login api will send token in request header. These api use POST method. There are following APIs :-

    1. *login API* :- Login api will be used by app which is used by all user to login.
        - Request :- Request will contain **email**, **password**.
        - Response :- Response will be **UserObject**.
        - Once we get UserObject then we app will get hostel from it and will populate subscription table.

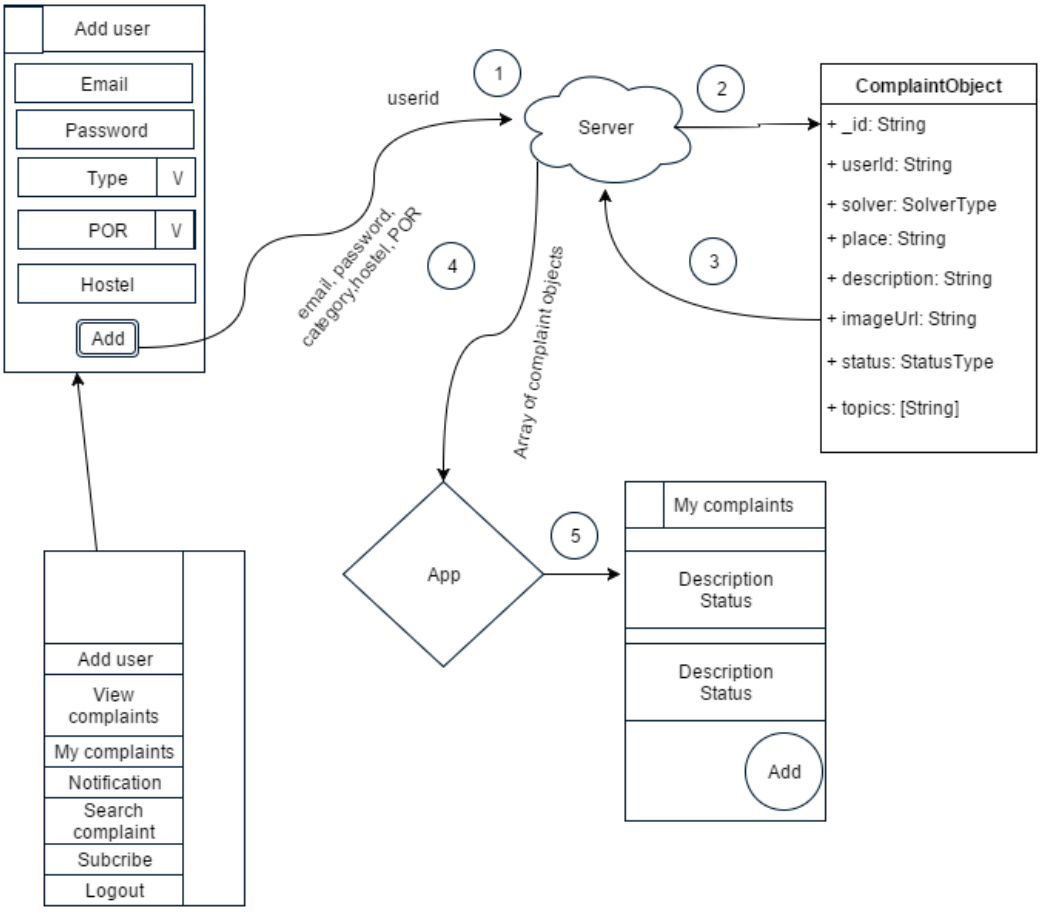


Figure 7: API 2

2. *create users API* :- Create user api will be used by app used by admin to add new user to database.
   - Request :- Request will contain **email**, **password**, **category** , **Hostel**.
   - Response :- Response will be **User Added/Not added**.

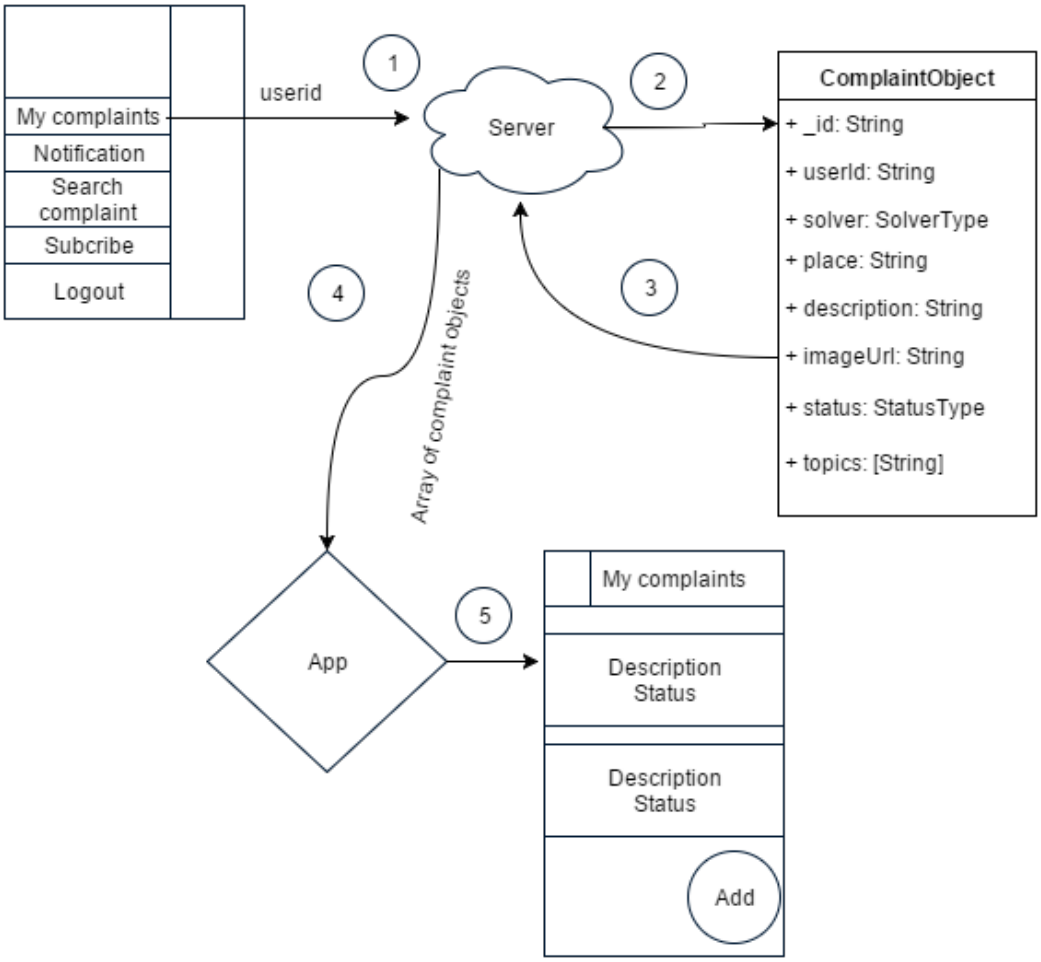


Figure 8: API 3

3. *my complaints API* :- My complaint api will be used by app which is used by admin and student to get complaint list.
   - Request :- Request will contain **userId**. App will get this user id from UserObject.

– Response :- Response will be **Array of ComplaintObjects**.

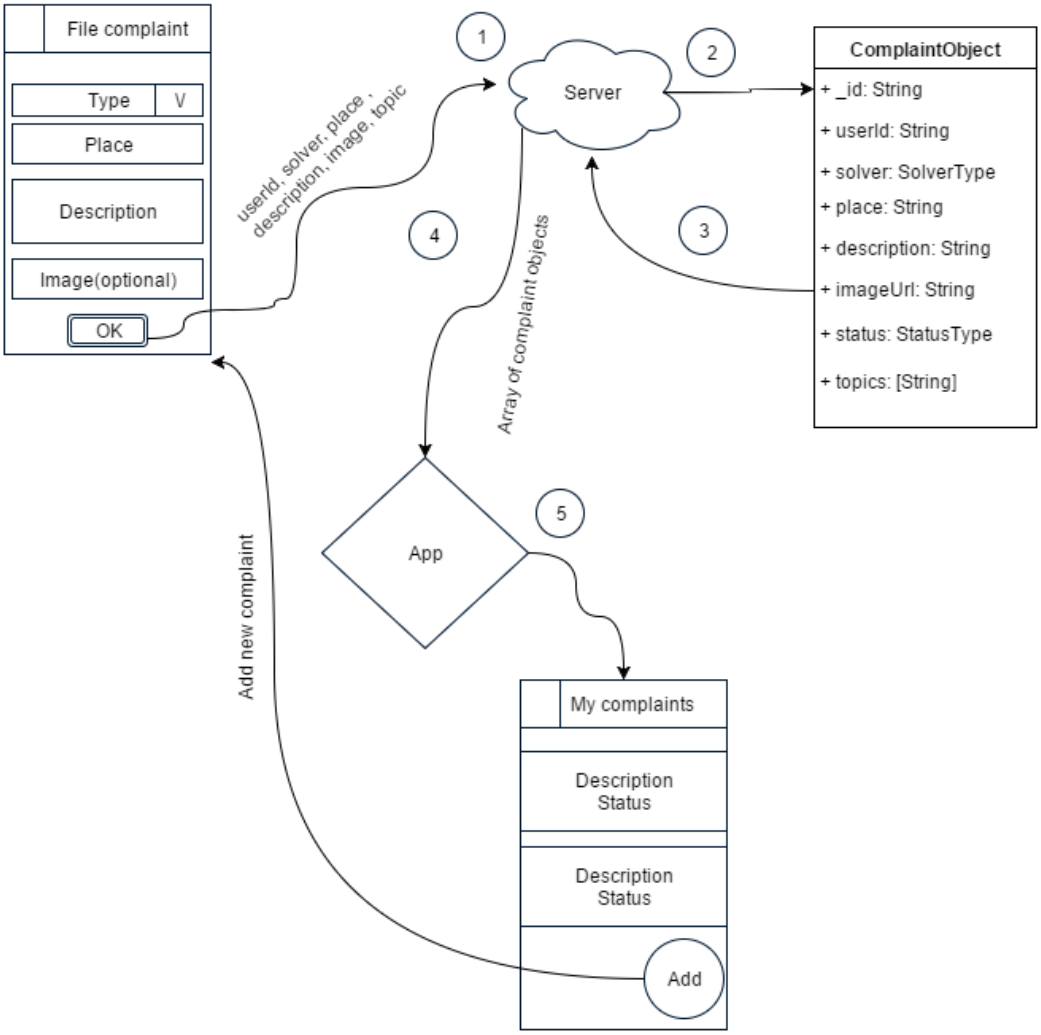


Figure 9: API 4

4. *file new complaint API* :- file new complaint api will be used by app which is used by admin and student to post new complaint.
   – Request :- Request will contain **userId**, **solver**, **place** , **description**, **image(file)**, **topic**. if Warden is solver then topic will be (hostel/...) or if Dean is solver then topic will be (insti).

– Response :- Response will be **ComplaintObject**.

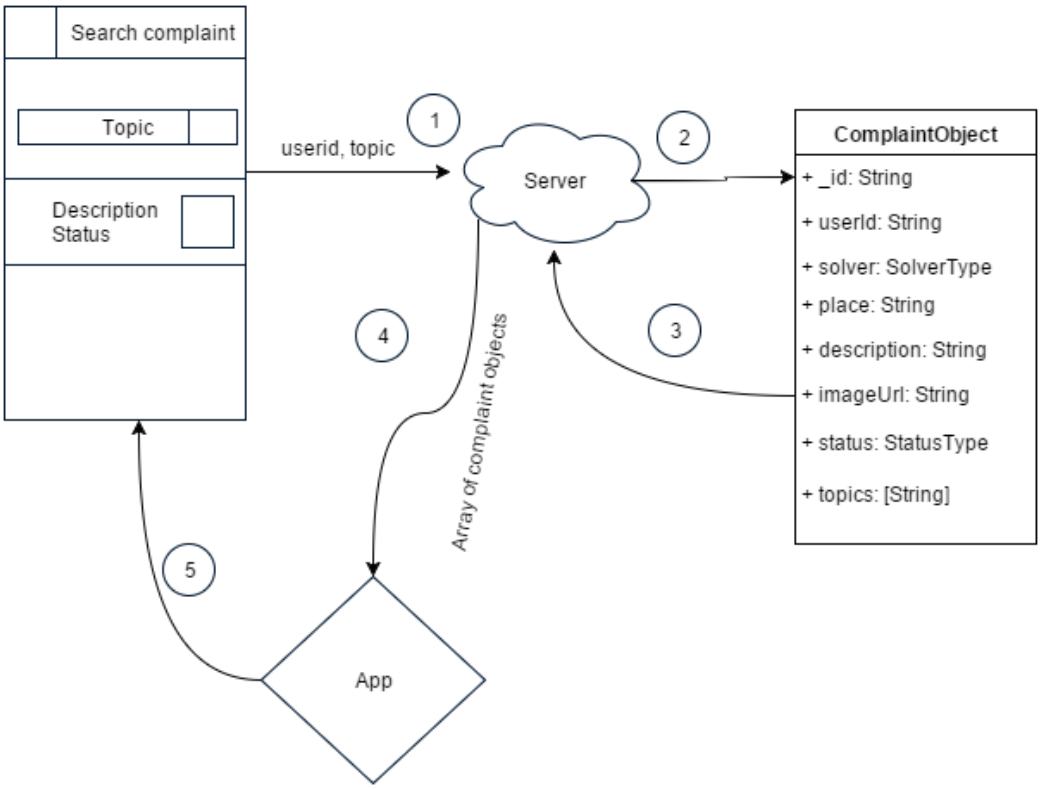


Figure 10: API 5

5. *search complaint API* :- search complaint api will be used by app which is used by all user to search a complaint if spefic type.
   - Request :- Request will contain **topic**.
   - Response :- Response will be **Array of ComplaintObject(for required topics)**.
6. *logout API* :- logout api will be used by app which is used by all user to logout. This will delete token from app and server then redirects to login screen.
7. *change complaint status API* :- change complaint status api will be used by app which is used by waden or dean to set status of complaint.

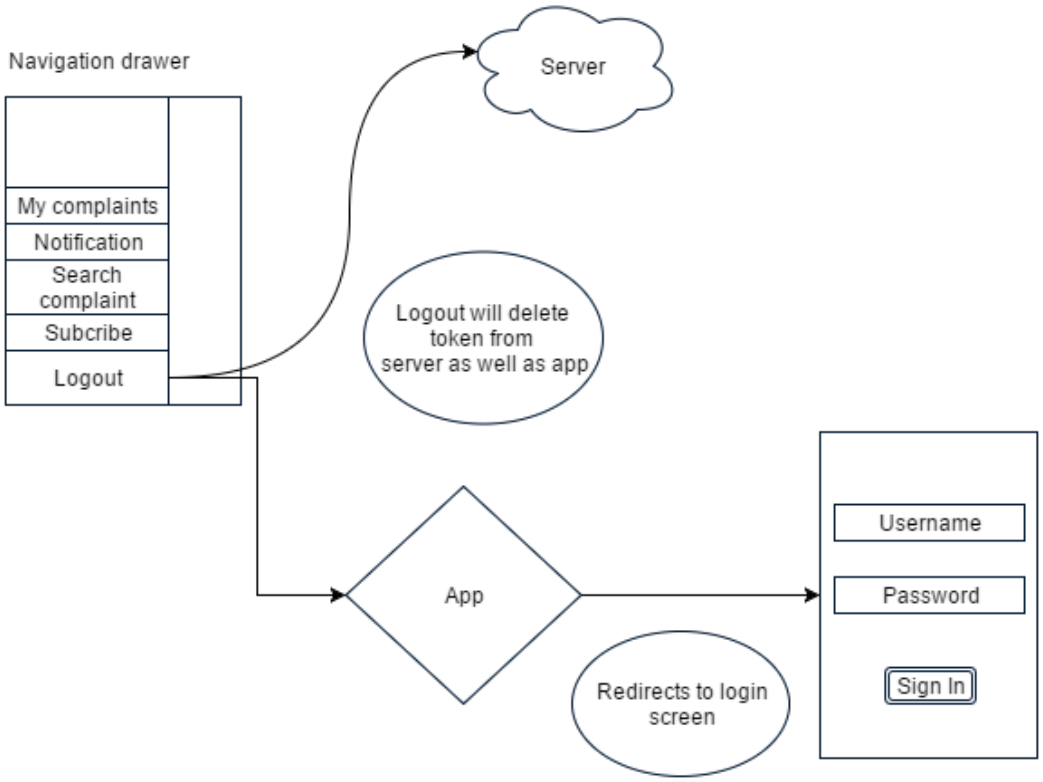


Figure 11: API 6

- Request :- Request will contain **complaint id**, **status**.
- Response :- Rosponse will be **"Sucess/ Failed"**, **status**.
- Notification of status will be sent to topics.

8. *particular complaint description API* :- this api will be used by app which is used by all user. When user clicks on item of list of complaints.
   - Request :- Request will contain **complaintId**.
   - Response :- Response will be **ComplaintObject**.
9. *getImage API* :- this api will be used by app which is used by all user to get image file.
   - Request :- Request will contain **complaintId**.

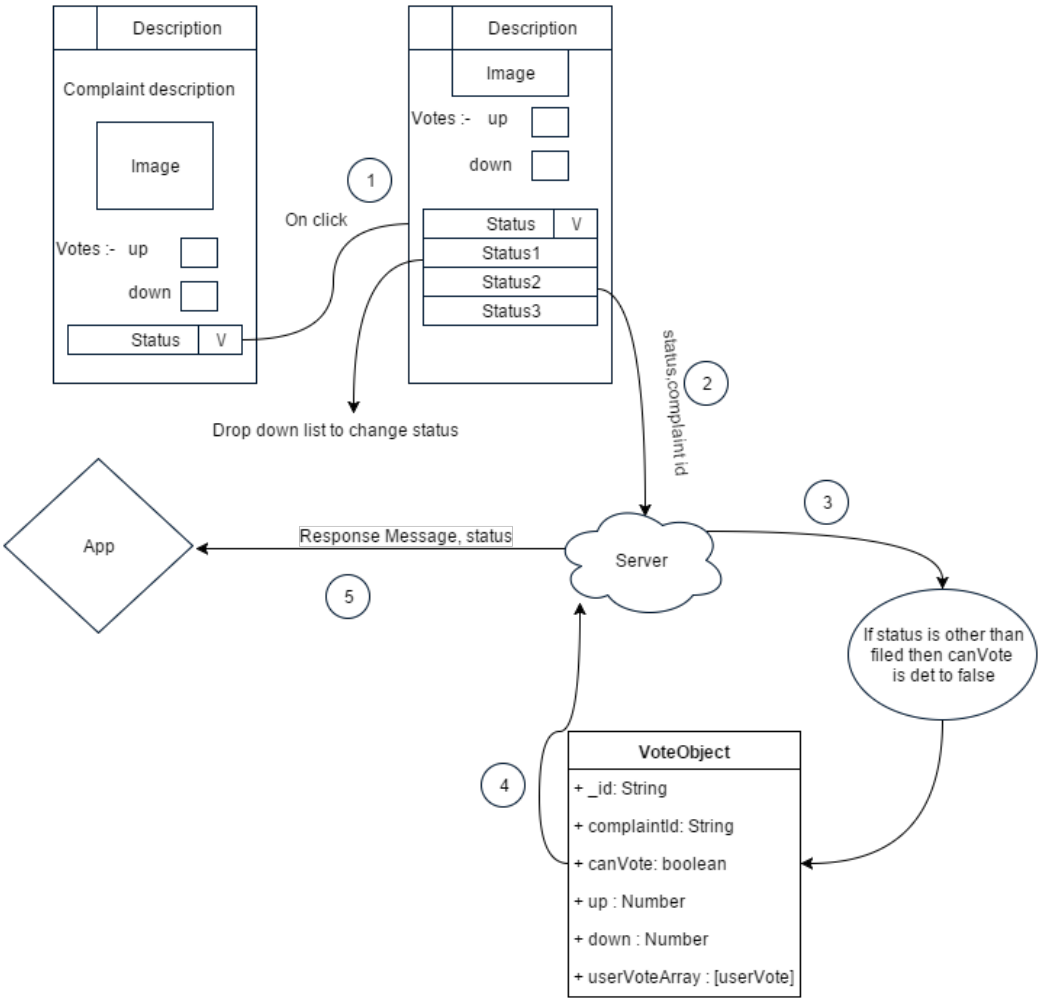


Figure 12: API 7

- Response :- Response will be **Image(from image url of ComplaintObject)**.

10. *vote API* :- this api will be used by app which is used by admin and student to vote for particular complaint.
    - Request :- Request will contain **complaintId**, **boolean(for up)**, **userId**.
    - Response :- Response will be Sucessful/ Try again.
    - if Sucessful is response then up/down ui will change.

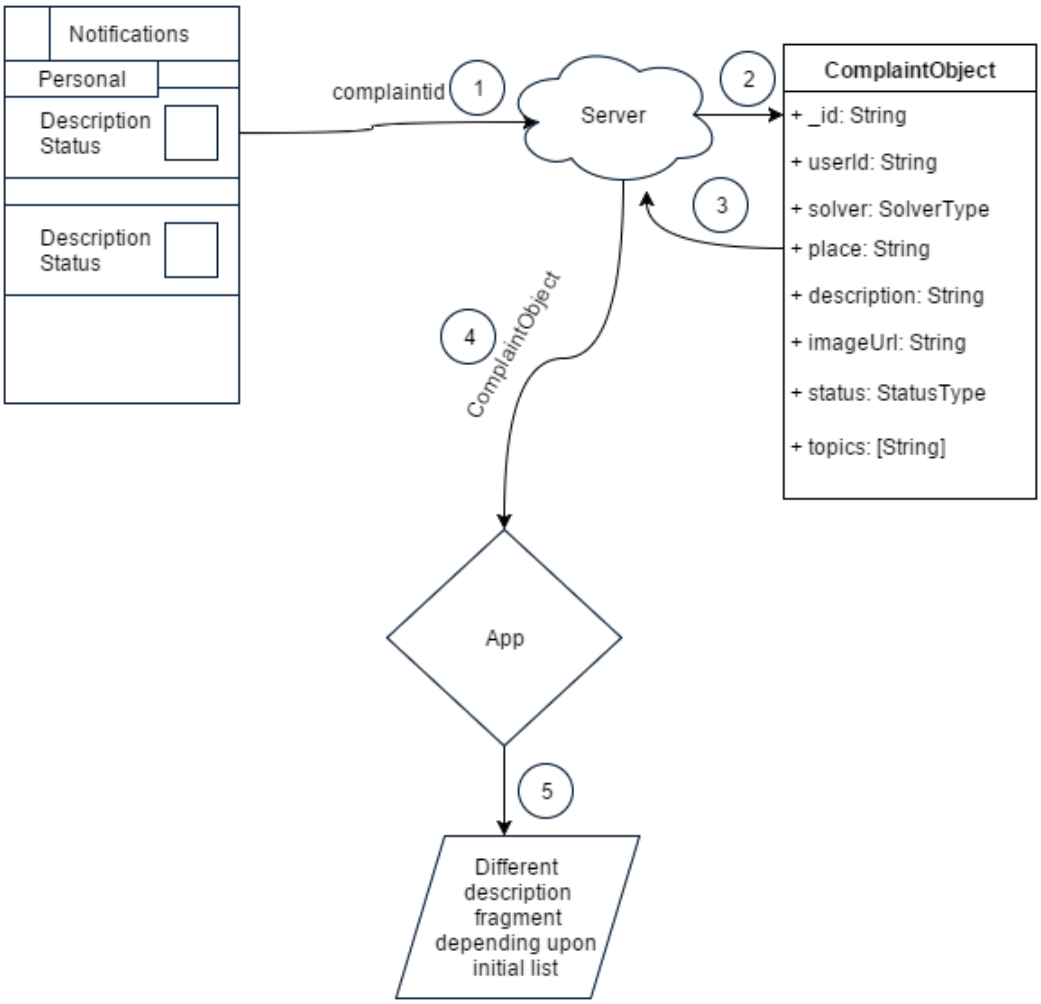


Figure 13: API 8

11. *change personal complaint status API* :- this api will be used by app which is used by admin and student to change status for personal complaint.
    - Request :- Request will contain **complaintId**, **Status**.
    - Response :- Response will be Sucessful/ Try again.

- So when user selects any item or any button corrosponding api will be used and after getting response the sebsequent fragment will get populated with information which app have received from server.

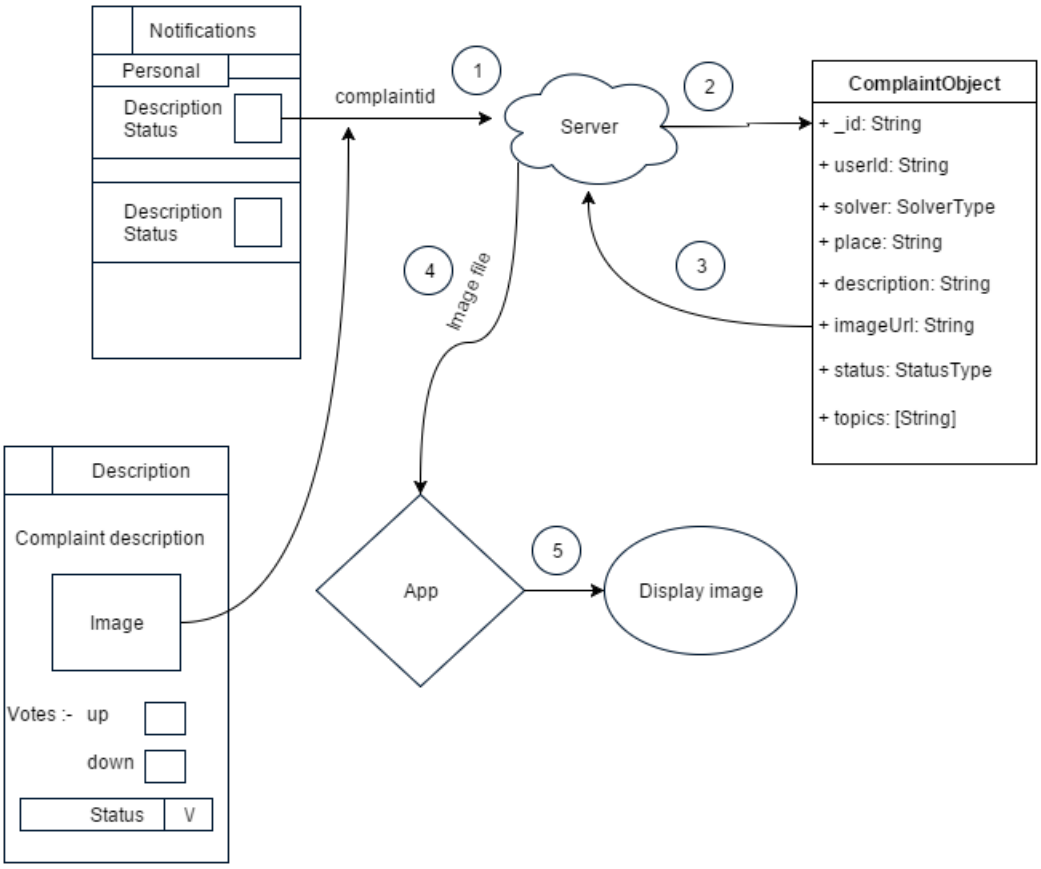


Figure 14: API 9

- For sending push notifications app uses MQTT (An open protocol designed for the Internet of things)
- Advantages of MQTT over GCM :-
  - Pub/Sub : A client can subscribe to multiple topics.
  - Quality of Service (QoS): GCM is pretty much "fire and forget". You receive acknowledgement that the message has been received but the server, but not that it is delivered to the client, so you will never know if the client got the message unless they communicate back to the server (which my code used to do). MQTT has 3 levels of QoS. Which vary from "fire and forget" to full acknowledgement that the client received the message.

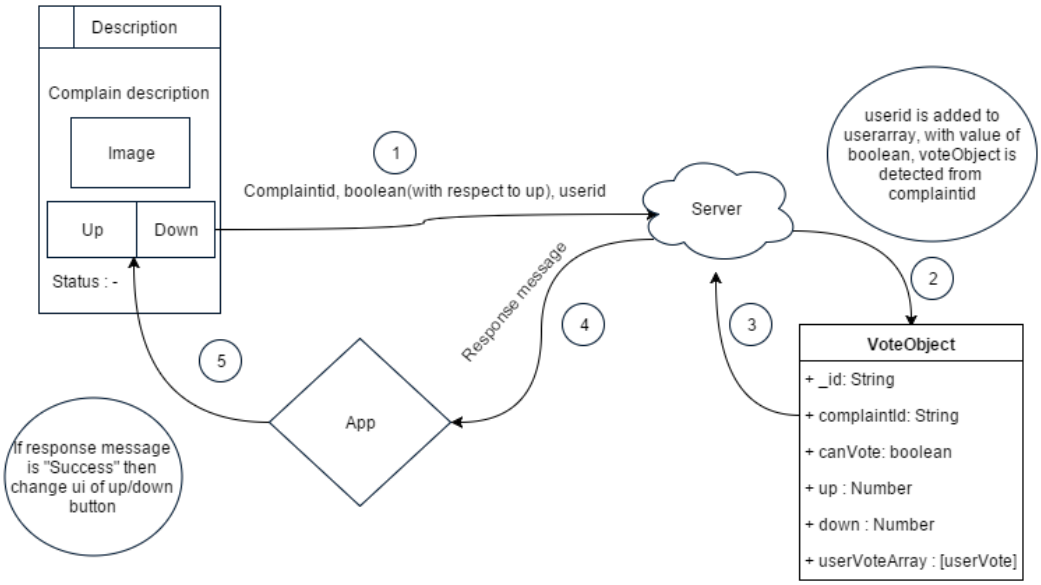


Figure 15: API 10

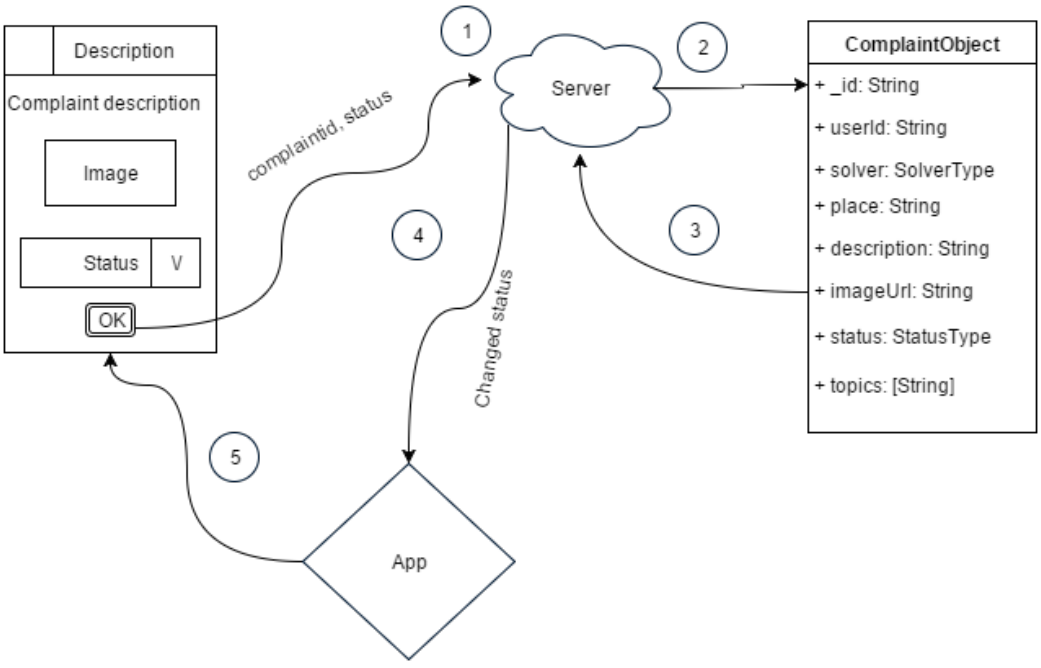
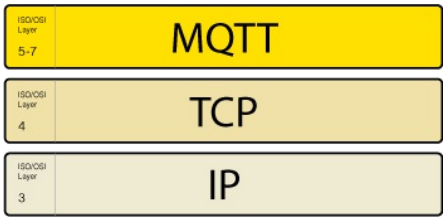
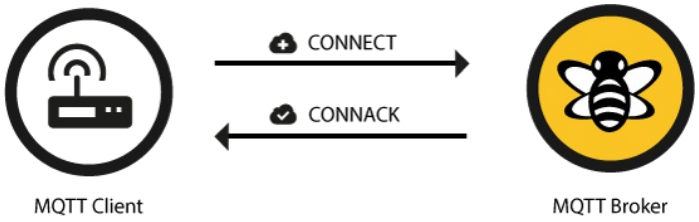


Figure 16: API 11

The MQTT protocol is based on top of TCP/IP and both client and broker need to have a TCP/IP stack.

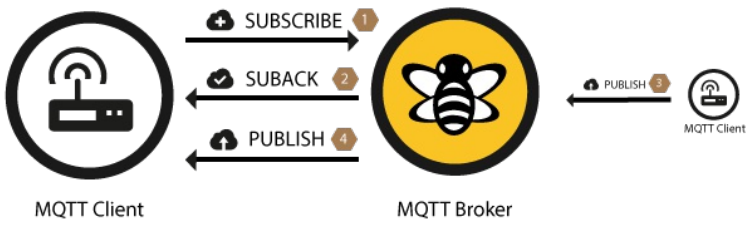


MQTT Client includes publisher or subscribers, both of them label an MQTT client that is only doing publishing or subscribing. (In general a MQTT client can be both a publisher & subscriber at the same time). A MQTT client is any device from a micro controller up to a full fledged server, that has a MQTT library running and is connecting to an MQTT broker over any kind of network.
The counterpart to a MQTT client is the MQTT broker, which is the heart of any publish/subscribe protocol. Depending on the concrete implementation, a broker can handle up to thousands of concurrently connected MQTT clients. The broker is primarily responsible for receiving all messages, filtering them, decide who is interested in it and then sending the message to all subscribed clients.It also holds the session of all persisted clients including subscriptions and missed messages

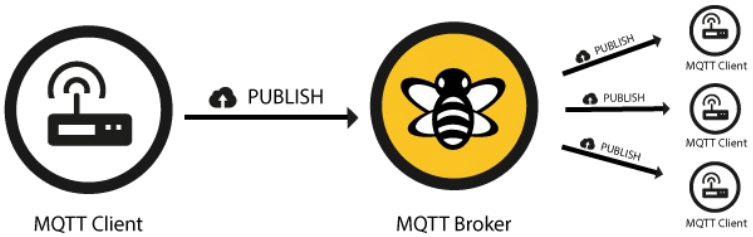


The MQTT connection itself is always between one client and the broker, no client is connected to another client directly. The connection is initiated through a client sending a CONNECT message to the broker. The broker response with a CONNACK and a status code. Once the connection is established, the broker will keep it open as long as the client doesn't send a disconnect command or it looses the connection.

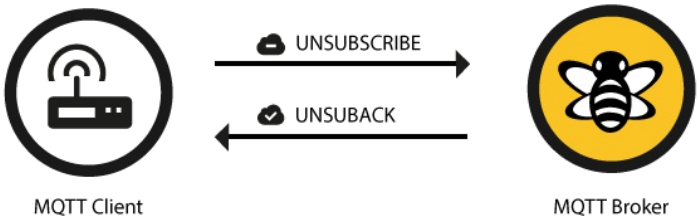
Figure 17: Connect server



After a client successfully sent the SUBSCRIBE message and received the SUBACK message, it will receive every published message matching the topic of the subscription



The client, who initially published the message is only concerned about delivering the publish message to the broker. From there on it is the responsibility of the broker to deliver the message to all subscribers. The publishing client doesn't get any feedback, if someone was interested in this published message and how many clients received the message by the broker.



After receiving the UNSUBACK from the broker, the client can assume the subscriptions in the UNSUBSCRIBE message are deleted.

Figure 18: Server