

COP 290 Assignment 2

Complaint Management System



Aditi

2014CS10205

Ayush Bhardwaj

2014CS10091

Nikhil Gupta

2014CS50462

March 2016

Contents

1	Objectives	3
2	Overall Design	3
3	User Interface	4
3.1	Front End	5
3.1.1	Splash Screen	5
3.1.2	Login Screen	5
3.1.3	Complaint List Screen	5
3.1.4	Complaint Type Screen	6
3.1.5	Complaint Information Screen	6
4	Features	6
4.1	Individual Complaints	6
4.1.1	Options for the user	6
4.1.2	Options for the receiver	7
4.1.3	Follow up options	7
4.1.4	Complaint Categories	7
4.2	Hostel Level Complaints	7
4.2.1	User Options	7
4.2.2	Other's Options	8
4.2.3	Receiver's Options	8
4.2.4	Follow Up options	8
4.2.5	Complaint Categories	8
4.3	Institute Level Complaints	8
4.3.1	User Options	9
4.3.2	Other's Options	9
4.3.3	Receiver's Options	9
4.3.4	Follow Up options	9
4.3.5	Complaint Categories	9
5	Sub Components	9
5.1	Server Back End	9
5.1.1	Databases	10
5.2	Network APIs	13
5.3	Further server tasks	16
5.4	Android App Back End	17
6	Interaction amongst Sub Components	18
6.1	API and Server	18
6.2	Internal Server	20

7	Testing Of Components	20
7.1	Server and APIs	20
7.2	Android App	20
7.3	Overall Testing	21
8	Extra Features	21
9	Future Endeavors	21
10	Source Code	21

1 Objectives

Design an App which is :

- Complaint Management System for IIT Delhi
- The Users of the App would be the people of the institute such as faculties, students, and institute employees
- Addresses three levels of complaints:
 1. Individual Complaint
 2. Hostel-level Complaint
 3. Institute-level complaint
- The users can submit their grievance to the the concerned authorities

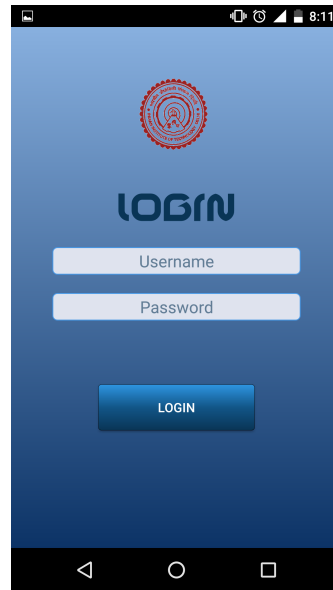
2 Overall Design

1. The server side will be programmed in Web2py [\[1\]](#)
2. The android app will be made using Java
3. The app would be extended to other phones as well as web clients
4. Volley will be used to send requests and receive responses
5. Shared Preferences will be used to maintain the feature of Persistent login
6. Doxygen will be used to create HTML documentation of the entire code base.
7. App would support multiple screen sizes [\[?\]](#)
8. Users can select their preferred topics.Only complaints related to selected topics will be visible to the user.
9. The entire code will be split up in multiple files to ensure modularity in code.

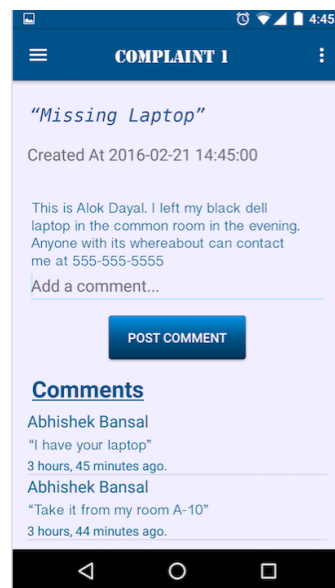
3 User Interface



(a) Splash Screen



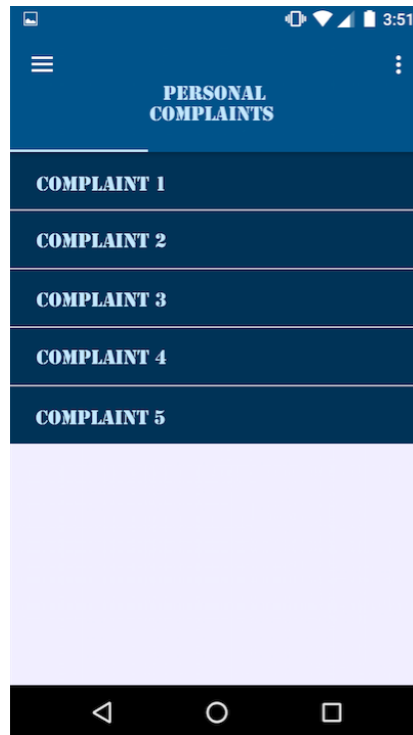
(b) Login Page



(a) Complaint Information UI



(b) Complaints List View



(a) Complaint Type Screen

3.1 Front End

3.1.1 Splash Screen

- This is the first screen that is displayed on the starting of the app.
- It has been made dynamic and interactive by adding animations and gesture recognitions

3.1.2 Login Screen

- The login page requests for the user's username and password and validates the login.
- It is displayed only after left swiping the splash screen.

3.1.3 Complaint List Screen

- This screen is a tabbed activity that contains the following tabs:
 1. Notifications (Displays new complaints)
 2. Complaints (Displays all the complaints, sorted by their types)

3. Alerts (Displays the complaints important to the user)

- The tabs contain information using ListViews

3.1.4 Complaint Type Screen

- This screen lists all complaints of the same type in an ExpandableList type.
- Clicking on a particular complaint will list some basic details about the complaint.
- Applying the sliding gesture on the complaint will open up the detailed complaint information screen.

3.1.5 Complaint Information Screen

- This screen contains all the indepth details of the complaint.
- It allows the user to add comments to the thread of the complaint.

4 Features

4.1 Individual Complaints

- The end user's complaint will be visible to the concerned authority only.
- Once the complaint is addressed,the user should be able to mark it as resolved
- The user can mark the complaint as resolved or take the complaint to the higher authority

4.1.1 Options for the user

The person filing the complaint can fill in the following options:

1. Name
2. Entry Number or Employee Number
3. Phone Number
4. Complain category
5. Address/Hostel Name
6. Content of the Complain
7. Extra details

4.1.2 Options for the receiver

1. See all details
2. Request user to mark as completed
3. Take to higher authority

4.1.3 Follow up options

1. Mark the complain as completed
2. Take to higher authority

4.1.4 Complaint Categories

1. Room Maintenance
2. Student Counselling Services
3. BSW
4. Academic

4.2 Hostel Level Complaints

- The end user's complaint would be visible to all the hostel residents who will be affected by the complaint.
- The concerned authority would have the right to mark the complaint as resolved.
- The complaint will be referred to a higher authority in a hierarchical order if
 1. Authority has marked the complaint as resolved but majority of users are not satisfied
 2. The time taken to resolve the complaint has exceeded a given time
- The Warden will be the highest authority. Veto powers will be given to the Warden

4.2.1 User Options

1. Name or Anonymous
2. Entry Number or Anonymous
3. Phone Number or Anonymous
4. Complain category
5. Contents
6. Additional Information

4.2.2 Other's Options

1. People can Upvote, Downvote or be neutral to the complain.
2. If the problem is not resolved in 7 days according to the majority, then the complain gets sent to the higher authority.
3. People can add comments to the complain.

4.2.3 Receiver's Options

1. Mark as resolved.
2. Send to higher authority

4.2.4 Follow Up options

1. Mark as resolved based on majority
2. Send to higher authority based on majority

4.2.5 Complaint Categories

1. Mess
2. Maintenance
3. Sports
4. BSW
5. Cultural Clubs

4.3 Institute Level Complaints

- The end user's complaint would be visible to all the institute residents who will be affected by the complaint.
- The concerned authority would have the right to mark the complaint as resolved.
- The complaint will be referred to a higher authority in a hierarchical order if
 1. Authority has marked the complaint as resolved but majority of the users are not satisfied
 2. The time taken to resolve the complaint has exceeded a given time
- The Concerned Dean will be the highest authority possessing Veto powers

4.3.1 User Options

1. Name or Anonymous
2. Entry Number or Anonymous
3. Phone Number or Anonymous
4. Complain category
5. Contents
6. Additional Information

4.3.2 Other's Options

1. People can Upvote, Downvote or be neutral to the complain.
2. If the problem is not resolved in 7 days according to the majority, then the complain gets sent to the higher authority.
3. People can add comments to the complain.

4.3.3 Receiver's Options

1. Mark as resolved.
2. Send to higher authority

4.3.4 Follow Up options

1. Mark as resolved based on majority
2. Send to higher authority based on majority

4.3.5 Complaint Categories

1. General Infrastructure Faults
2. Security
3. Network Connectivity
4. Cultural and Technical Clubs/Organisations
5. Academic

5 Sub Components

5.1 Server Back End

The back end has been divided into further sub components to facilitate the development process.

5.1.1 Databases

Table 1: User Database Table

S.No.	Fields	Type	Description
1	Name	String	Name of the person
2	Unique Id	String	Entry Number for students or Employee Code for staff and faculty
3	User Type	Int	Information regarding user being a student or staff or faculty
4	Contact Number	String	Phone Number
5	Hostel	String	Hostel if any
6	Other Details	String	Any other details
7	Password	String	Password in hashed form
8	User Id	String	Entry Number for students or Employee Code
9	Hostel Preferences	String	Bit string to represent interest in Hostel activities
10	Institute Preferences	String	Bit string to represent interest in Institute activities
11	Extra Preferences	String	Bit String to represent interest in Other activities

Table 2: Administrator Information

S.No.	Fields	Type	Description
1	User Id	String	Unique user id
2	Complaint Area	Int	Integer code for complaint area
3	Level	Int	Priority level for person
4	Description	String	Extra information
5	Hostel Id	Int	ID of hostel if any

Table 3: Responses to a Complaint

S.No.	Fields	Type	Description
1	User Id	String	Entry Number for students or Employee Code for staff and faculty
2	Complaint Id	Int	Integer code for complaint area
3	Response	Int	Int corresponding to Upvote, Downvote or being neutral

Table 4: Hostel Mapping

S.No.	Fields	Type	Description
1	Hostel Id	Int	Integer for a hostel
2	Hostel Name	String	String for hostel

Table 5: Complaint Category Mapping

S.No.	Fields	Type	Description
1	Category Id	Int	Integer for a complaint category
2	Category Description	String	String for a complaint category

Table 6: Comments on a complaint

S.No.	Fields	Type	Description
1	Complaint Id	String	Unique Id for Complaint
2	User Id	String	Unique User Id
3	Description	String	Complaint comment
4	Time Stamp	Time	Time of Comment
5	Anonymous	Boolean	Whether user is anonymous

Table 7: Complaint User Mapping

S.No.	Fields	Type	Description
1	Complaint Id	String	Unique Id for Complaint
2	User Id	String	Unique User Id

Table 8: Notifications

S.No.	Fields	Type	Description
1	Complaint Id	String	Unique Id for Complaint
2	From User Id	String	Unique User Id
3	To User Id	String	Unique User Id
4	Description	String	Content of notification
5	Seen	Boolean	Marked as seen or not

Table 9: User Satisfaction Response

S.No.	Fields	Type	Description
1	User Id	String	Entry Number for students or Employee Code for staff and faculty
2	Complaint Id	Int	Integer code for complaint area
3	Response	Int	Int corresponding to Satisfied or not

Table 10: Individual Complaint

S.No.	Fields	Type	Description
1	Complaint Id	String	Unique Id for Complaint
2	User Id	String	Unique User Id
3	Complaint Type	Int	Complaint category
4	Complaint Content	String	Content of complaint
5	Extra Info	Image	Upload a photo
6	Admin Id	String	Id of person assigned
7	Time Stamp	Time	Time of filing the complaint
8	Resolved	Boolean	Resolved or Not
9	Mark for resolution	Boolean	Option for complaint addressee to seek approval
10	Comment	String	Any comments
11	Previous Id	Int	Previous complaint id if any

Table 11: Hostel Level Complaint

S.No.	Fields	Type	Description
1	Complaint Id	String	Unique Id for Complaint
2	User Id	String	Unique User Id
3	Complaint Type	Int	Complaint category
4	Complaint Content	String	Content of complaint
5	Extra Info	Image	Upload a photo
6	Admin Id	String	Id of person assigned
7	Time Stamp	Time	Time of filing the complaint
8	Resolved	Boolean	Resolved or Not
9	Mark for resolution	Boolean	Option for complaint addressee to seek approval
10	Comment	String	Any comments
11	Previous Id	Int	Previous complaint id if any
12	Hostel	Int	Hostel Id
13	Anonymous	Boolean	Anonymous or not
14	Number of Up-votes	Int	Number of Up-votes
15	Number of Down-votes	Int	Number of Down-votes
16	Number of Neutrals	Int	Number of Neutral people
17	Number of Satisfied	Int	Number of people satisfied
18	Number of Dissatisfied	Int	Number of people dissatisfied with solution

Table 12: Institute Level Complaint

S.No.	Fields	Type	Description
1	Complaint Id	String	Unique Id for Complaint
2	User Id	String	Unique User ID
3	Complaint Type	Int	Complaint category
4	Complaint Content	String	Content of complaint
5	Extra Info	File	Upload a photo
6	Admin Id	String	Id of person assigned
7	Time Stamp	Time	Time of filing the complaint
8	Resolved	Boolean	Resolved or Not
9	Mark for resolution	Boolean	Option for complaint addressee to seek approval
10	Comment	String	Any comments
11	Previous Id	Int	Previous complaint id if any
12	Anonymous	Boolean	Anonymous or not
13	Number of Up-votes	Int	Number of Up-votes
14	Number of Down-votes	Int	Number of Down-votes
15	Number of Neutrals	Int	Number of Neutral people
16	Number of Satisfied	Int	Number of people satisfied
17	Number of Dissatisfied	Int	Number of people dissatisfied with solution

5.2 Network APIs

The following API's will be provided: [?]

1. **Login**
URL: `/default/login.json`
Returns: JSON object with Success/Failure and UserDetails
Request: POST containing UserName And Password
2. **Logout**
URL: `/default/logout.json`
Returns: JSON object with Success/Failure
Request: GET
3. **Change Password**
URL: `/default/change_pass.json`
Returns: JSON object with Success/Failure
Request: POST containing the changed password
4. **Get Notifications**
URL: `/default/notifications.json`
Returns: JSON object containing list of all notifications.
Request: GET
5. **Get Preferences**
URL: `/user/preferences.json`

Returns: JSON object containing bitstrings of preferences and corresponding preference areas.

Request: GET

6. **Set Preferences**

URL: `/user/update_preferences.json?hostel=<hostel_pref>&institute=<insti_pref>&extra=<extra_pref>`

Returns: JSON object containing updated bitstrings and corresponding preference areas.

Request: GET

7. **Get Administrator**

URL: `/admin/get_admin.json?complian_area=<Complain>&level=<level>&Hostel=<hostel_id>`

Returns: JSON object containing the administrator of a Complaint at a particular level

Request: GET

8. **Get Response**

URL: `/complaint/get_user_response.json?complian_id=<Complain>`

Returns: JSON object containing the response of a user to a public complaint

Request: GET

9. **Set Response**

URL: `/complaint/set_user_response.json?complian_id=<Complain>&Response=<response>`

Returns: JSON object with Success/Failure.Sets the response of a user to a public complaint

Request: GET

10. **Complain Category Mapping**

URL: `/complaint/get_desc.json?category_id=<cat>`

Returns: JSON object containing the category description

Request: GET

11. **Hostel Mapping**

URL: `/default/get_hostel.json?hostel_id=<hos_id>`

Returns: JSON object containing the Hostel,ID pair

Request: GET

12. **Get Comment**

URL: `/complaint/get_comments.json?complian_id=<id>`

Returns: JSON object containing all the comments on the thread to a complain

Request: GET

13. **Post Comment**

URL: `/complaint/post_comments.json?complian_id=<id>`

Returns: JSON Success Response on posting a comment on the thread to a complain
Request: GET

14. **Get Notification**

URL: `/notification/get_noti.json`

Returns: JSON object containing the set of notifications concerning the user

Request: GET

15. **Set Notification Status**

URL: `/notification/set_noti_status.json??compliant_id=<complain_id>`

Returns: JSON object with Success/Failure.Sets the seen/unseen status of the user

Request: GET

16. **Get User Satisfaction**

URL: `/complaint/get_user_satus.json?compliant_id=<complain_id>`

Returns: JSON object containing the user's satisfaction after the complaint is resolved

Request: GET

17. **Update user response**

URL: `/complaint/put_user_satus.json?compliant_id=<complain_id>&response<bool>`

Returns: JSON object with Success/Failure.Sets the user's satisfaction after the complaint is resolved

Request: GET

18. **Add Complaint**

URL: `/complaint_data/add_complaint.json`

Returns: JSON object with Success/Failure.Adds a complaint to the database

Request: POST containing complain type,complaint content,extra.info,anonymous.boolean

19. **Edit Complaint**

URL: `/complaint_data/edit_complaint.json`

Returns: Edits an existing complaint in the database

Request: POST containing complain type,complaint content,extra.info,anonymous.boolean

20. **Send to Higher Authority**

URL: `/complaint/hig_auth.json?complaint_id=<id>`

Returns: Sends the complaint to Higher Authority

21. **Get all Complaints** URL: `/complaint_data/get_all.json`

Returns: Return all the complaints concerning the user Request: GET

22. **Get Complaint Detail**
URL: `/complaint_data/get_complaint_details.json?complaint_id=<id>`
Returns: JSON object containing the details of a complaint
Request: GET
23. **Get Complaints to be Addressed**
URL: `/admin_complaint/get_all_complaints.json`
Returns: JSON object containing all the complaints to be addressed
Request: GET
24. **Get Complainant Details**
URL: `/admin_complaint/get_complainant.json?complaint_id=<id>`
Returns: JSON object containing the Complainant's detail
Request: GET
25. **Mark Approved**
URL: `/admin_complaint/mark_resolved.json?Complaint_id=<id>&boolean_resolve=<bool_done>`
Returns: JSON object with Success/Failure.Success in marking for the Complainant's Approval
Request: GET
26. **Add Administrator's Comment**
URL: `/admin_complaint/add_comment.json?complaint_id=<id>&desc=<description>`
Returns: JSON object with Success/Failure.Success in marking for the Complainant's Approval
27. **Take to Higher Authority**
URL: `/admin_complaint/take_to_higher.json?complaint_id=<id>`
Returns: JSON object with Success/Failure.Success in sending the complaint to higher authority
Request: GET
28. **Add User**
URL: `/default/add_user`
Returns: JSON object containing the details of user added to the database
Request: POST containing all the details of user

5.3 Further server tasks

1. Crontab tasks will be used to schedule periodic events like database cleanup and sending complaints to higher authorities.
2. Crontab will also be used to generate push notifications for devices.

5.4 Android App Back End

- Volley will be used to send requests and receive responses.
- A separate class will be used to maintain the Session Cookies.

Listing 1: App-Cookie Class

```
1 class App_Cookie extends Application
2 {
3     CookieManager cookieManage;
4     // Manages the Cookie Session for the Entire Application
5     public:
6     public void onCreate()
7     {
8         cookieManage = new CookieManager();
9         CookieHandler.setDefault(cookieManage);
10        super.onCreate();
11    };
12};
```

- Shared Preferences

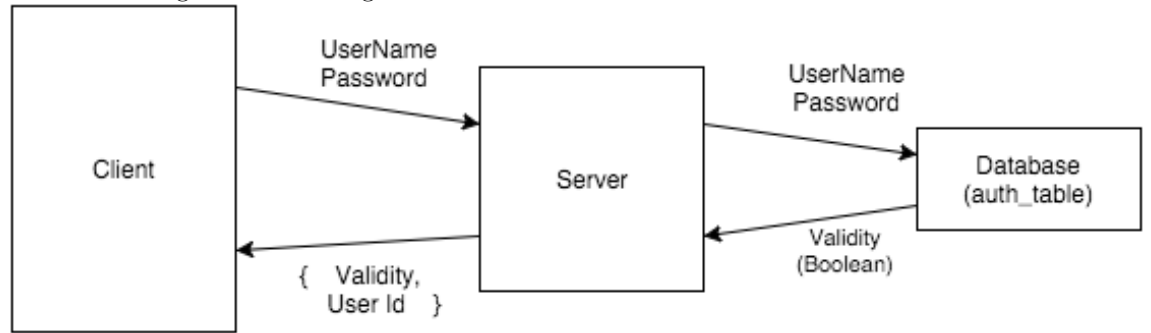
Listing 2: Persistent Login

```
1 EditText username, password;
2 public static final String AppPREFERENCES = "AppPrefs";
3 public static final String UserId = "userId";
4 SharedPreferences sharedPreferences;
5 public void check_logged_in()
6 { // Checks if the user is already logged.
7   // If logged in, he is directed to the HomePage.
8   sharedPreferences = getSharedPreferences
9       (AppPREFERENCES, Context.MODE_PRIVATE);
10  SharedPreferences.Editor editor = sharedPreferences.edit();
11  int user_id= sharedPreferences.getInt("userId", 0);
12  if(user_id!=0) // User_id is zero, if none is logged in
13  {   Intent i = new Intent(Login.this,HomePage.class);
14      i.putExtra("userId",user_id);
15      startActivity(i);
16  } // Logged in user is directed to his Home Page
17  }
18  return;
19};
```

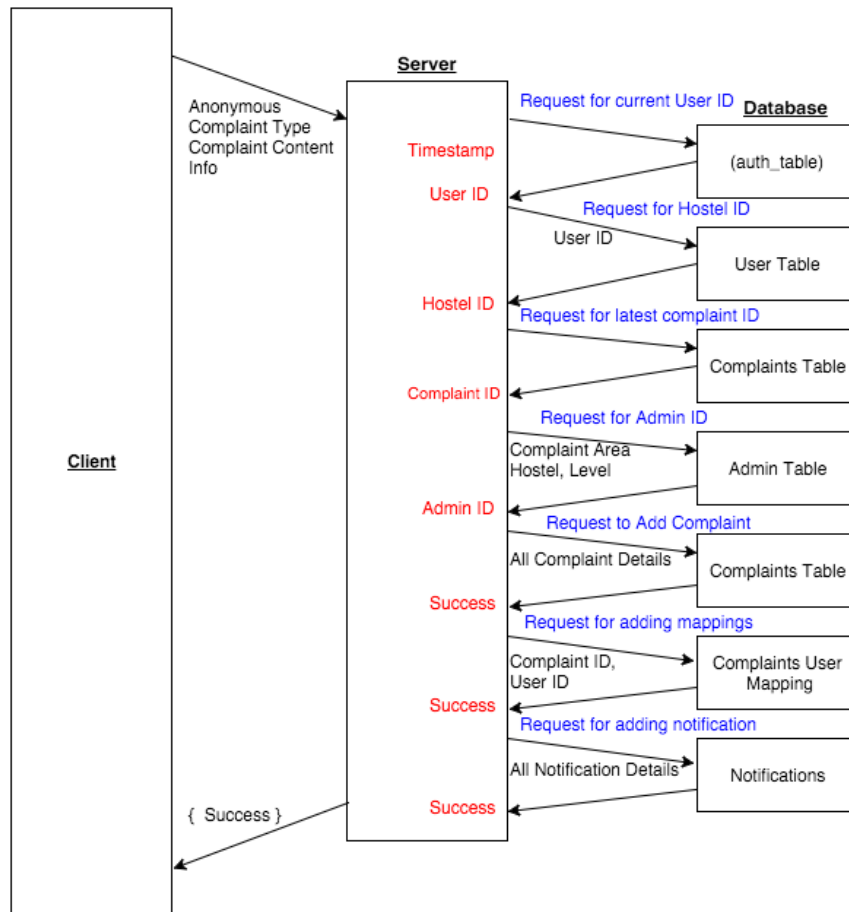
6 Interaction amongst Sub Components

6.1 API and Server

1. Event flow diagram for user login

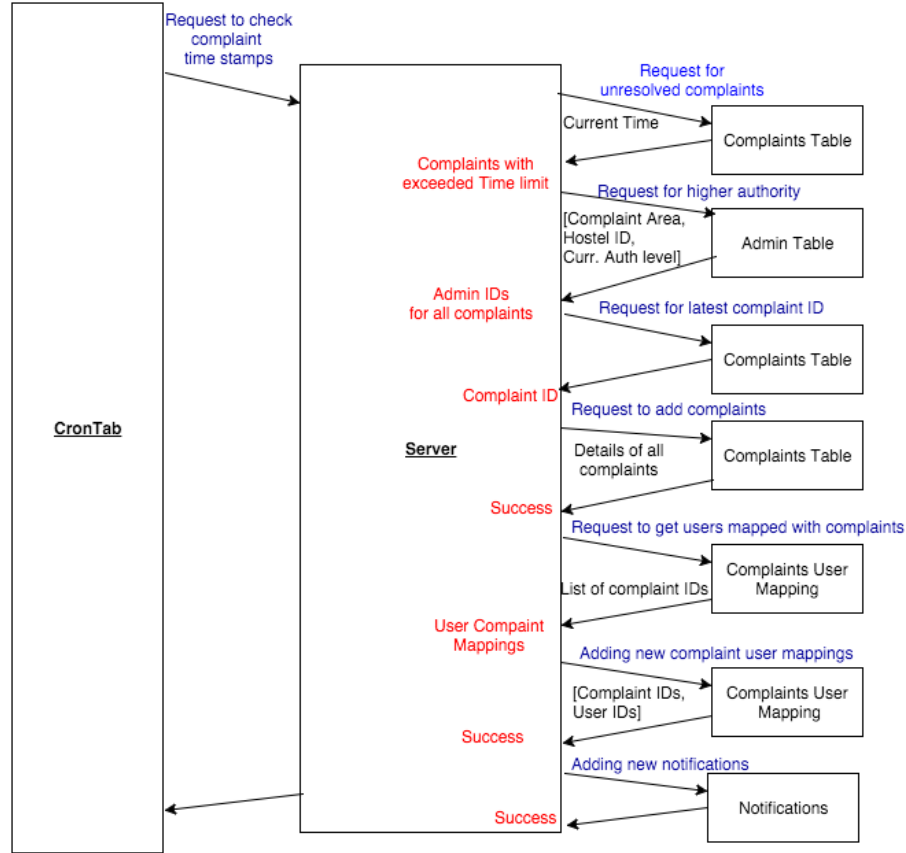


2. Event flow diagram for user adding a complaint.



6.2 Internal Server

Event flow diagram for crontab scheduling.



7 Testing Of Components

7.1 Server and APIs

- Unit testing will be used to check if the server and end points are working correctly.
- For each endpoint, stress testing will be done via python or bash scripts to verify that the APIs perform as expected in various situations.

7.2 Android App

We will use the Unit Testing component of Android studio to test each class created in the app for correctness. By doing this we will ensure that no errors take place on the android app and the user has a bug free experience.

7.3 Overall Testing

We will use the app on our phones once it is ready to identify and squash any remaining bugs in the app or the server.

8 Extra Features

- The scope of the Application will be extended to Web Clients
- Hierarchy of Authorities would be maintained for every Complaint Type and Category
- In public level complaints option to register complaint as anonymous would be provided
- Users would be allowed to select the categories that affect them. Only public level complaints related to these categories will be displayed to the user
- Notification would be marked seen/unseen
- Users will get option to mark as satisfactory/unsatisfactory after the complaint is resolved
- Users will be able to post comments in the thread related to the concerned complaint
- The admin has an option to make a phone call or send an email through the app to the person who lodged the complaint.

9 Future Endeavors

- Keep local cache of changes done, at mobile level and sync them with the global server as soon as Internet connectivity is supplied.
- To extend the scope of Application to Windows as well as iOS client. Same Networking API's will be used.
- Users to be synchronized into the database via their Kerberos ID
- View location in google map/instiapp

10 Source Code

The source code of the project is maintained in the following repository:

https://github.com/aditi741997/COP290_Assignment_2.git

References

- [1] Supporting tab and phone view. <http://developer.android.com/training/multiscreen/screensizes.html>.
- [2] Web2py. <http://www.web2py.com/appliances/>.