COP290: Complaint Management App

Ajmeera NagaRaju (2014CS10208)
Vipparthy Sai Esvar (2014CS10266)
Harshil R. Meena (2014CS10222)

April 9, 2016

1 Introduction

- The App will have the following features (corresponding APIs would be called for their implementation):
  - Login : as student or (warden/faculty or specialuser)
- After Login the user can:
  - View the list of complaints related to the user
  - View the list of notifications
  - Lodge a new complaint
  - Mark their vote on a common complaint
  - Mark the complaints in their jurisdiction as resolved
  - Logout
- Also, the specialuser can:
  - Add any new user to the database
- We will be using a public Web2Py server coded in Python for our application

- We have also assigned one account to the superuser who will be in-charge of adding other users. His login details are :
  - Username: superuser
  - Password: godmode
- The APIs will be implemented in such a manner that the assignment can be easily extended to the web clients as well

2 User Interface

- A blank activity for the login page with two entry fields username, password and a login button
- A navigation drawer activity for the main dashboard allowing the user the following options once he is successfully logged in:
  - List of all complaints
  - Lodge a complaint
  - View my complaint
  - Logout
- The fragment displaying the list of all complaints, serving as the home page when a user successfully logs in, consists of all the complaints which have been lodged by any of the users
- On clicking up on Lodge a complaint, the user will be allowed to put up his own complaint after switching to a new activity providing the following input fields:
  - Title
  - Person to whom the complaint is addressed to
  - Complaint Description
  - Complaint location and level
  - Submit button

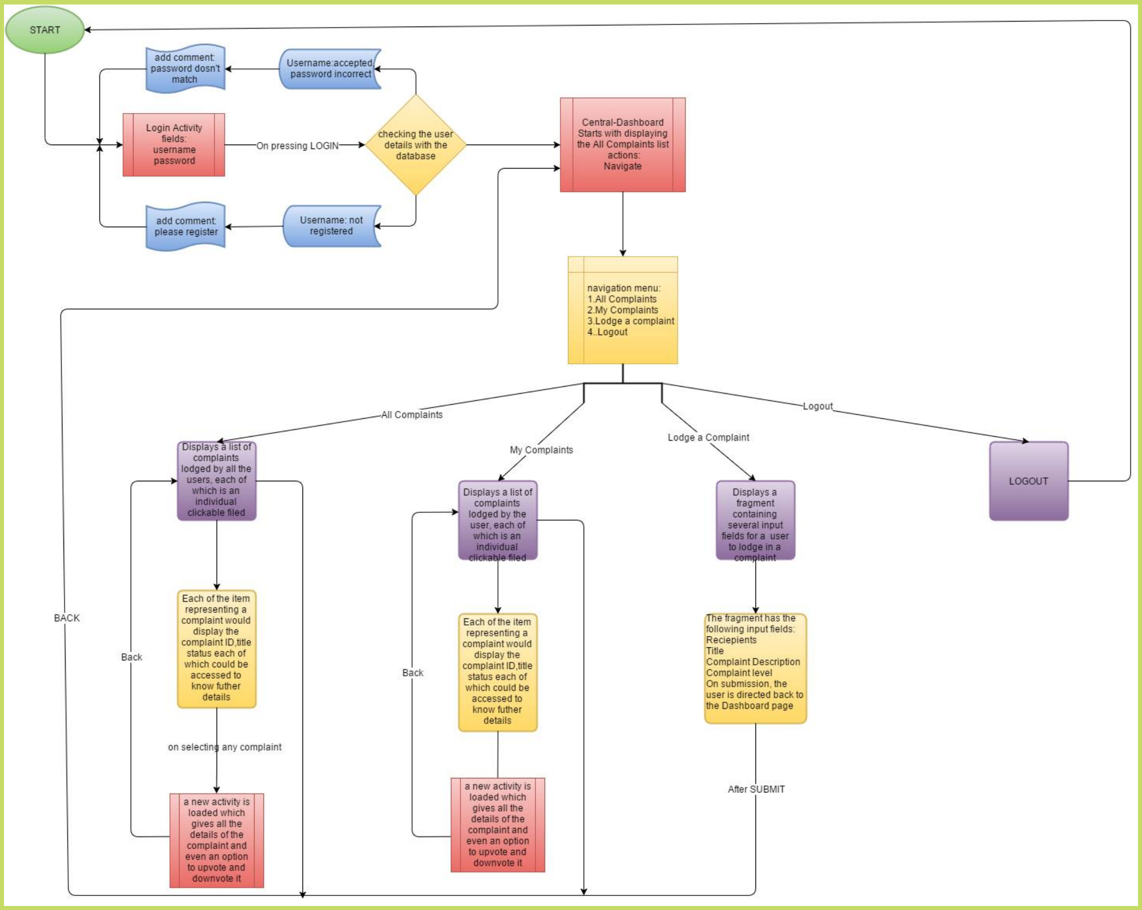
START
add comment: password doesn't match
Username:accepted password incorrect
Login Activity fields: username password
On pressing LOGIN
checking the user details with the database
Central-Dashboard Starts with displaying the All Complaints list actions: Navigate
add comment: please register
Username: not registered
navigation menu: 1.All Complaints 2.My Complaints 3.Lodge a complaint 4..Logout
All Complaints
My Complaints
Lodge a Complaint
Logout
Displays a list of complaints lodged by all the users, each of which is an individual clickable filed
Displays a list of complaints lodged by the user, each of which is an individual clickable filed
Displays a fragment containing several input fields for a user to lodge in a complaint
LOGOUT
BACK
Back
Each of the item representing a complaint would display the complaint ID,title status each of which could be accessed to know futher details
The fragment has the following input fields: Reciepients Title Complaint Description Complaint level On submission, the user is directed back to the Dashboard page
on selecting any complaint
After SUBMIT
a new activity is loaded which gives all the details of the complaint and even an option to upvote and downvote it

Figure 1: Workflow

- Once a complaint is submitted, a unique complaint ID will be generated for it and the user would be given an option to either return to the home page or lodge another complaint. Also,the complaint will be displayed in the list containing all the complaints as a fragment containing the complaint ID, title, location, status, options to either upvote or downvote
- If the user wishes to to go through the complaints lodged by him, he could choose the option from the navigation drawer which if for displaying his complaints
- On clicking on any of these complaints, the user can see:

- Title of the complaint
- Complaint ID
- Description of the complaint
- Level of complaint
- Number of upvotes/downvotes
- Also options to upvote and downvote
- Person to whom the complaint is addressed to

3 Event flow

- Login:
  - User enters the login details on the application and presses ENTER
  - Users details gets checked
  - POST request sent to the server in form of a StringRequest
  - The server then calls the Login API with the form data
  - The API checks if the login details are present in the database by fetching the content using SQLite
  - If the content is present in the tables then the Dashboard for the user gets shown
- Logout:
  - User presses the LOGOUT button
  - GET request sent to the server in form of a StringRequest
  - The server then calls the Logout API url
  - The user gets redirected to the Login screen
- Notifications:
  - User presses the NOTIFICATIONS button
  - GET request sent to the server in form of a StringRequest

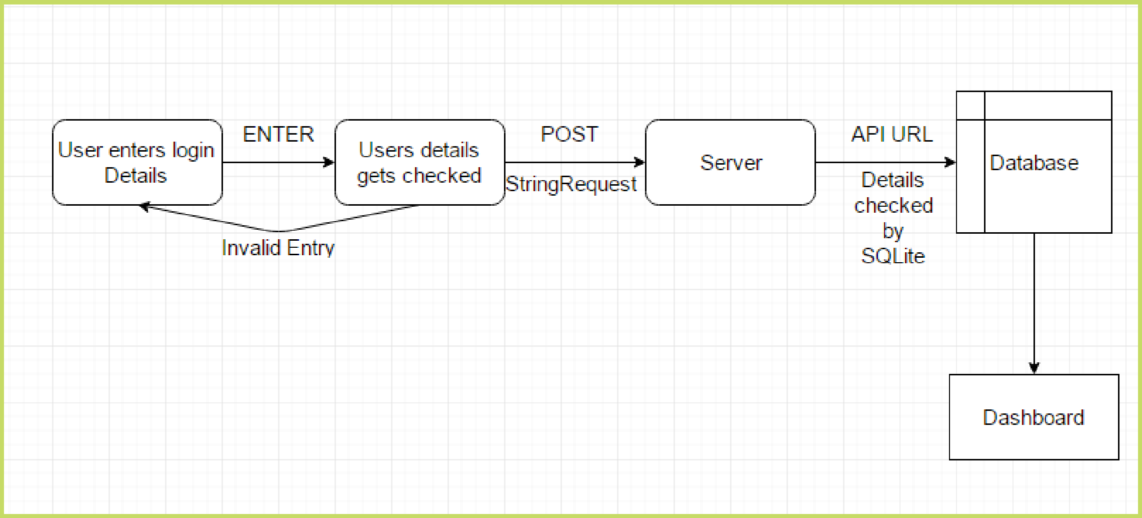
User enters login Details
ENTER
Users details gets checked
POST
StringRequest
Server
API URL
Details checked by SQLite
Database
Invalid Entry
Dashboard

Figure 2: Sample event flow

  - The server then calls the Notifications API url
  - The API fetches all the data from the Notifications Table using SQLite and sets it as the input for the ListView
  - The user gets redirected to the Notifications screen

- All complaints:
  - User presses the All complaints button
  - GET request sent to the server in form of a StringRequest
  - The server then calls the All complaints API url
  - The API fetches all the data from the Complaints Table using SQLite and sets it as the input for the ListView
  - The user gets redirected to the All complaints screen

- Vote on a complaint:
  - User presses the Upvote/Downvote button
  - POST request sent to the server in form of a StringRequest
  - The server then calls the All complaints API url with the vote data

  - The API fetches the updates the vote data of the complaint in the Complaints Table using SQLite and resets the input for the ListView
  - All complaints screen get reloaded

- Mark the complaint as resolved:
  - User presses the RESOLVED button
  - GET request sent to the server in form of a StringRequest
  - The server then calls the Mark the complaint as resolved API url with the complaint ID
  - The API updates the resolved data of the corresponding complaint in the Complaints Table using SQLite and resets the input for the ListView
  - All complaints screen gets reloaded

- Lodge a new complaint:
  - User enters the complaint title and description and then presses the LODGE button
  - POST request sent to the server in form of a StringRequest
  - The server then calls the Lodge a new complaint API url with the complaint data
  - The API inserts this complaint into the Complaint table using SQLite
  - Lodge a new complaint screen get reloaded

- Add new user to db:
  - superuser enters the user details and presses ENTER button
  - POST request sent to the server in form of a StringRequest
  - The server then calls the Add new user to db API url with the user data
  - The API inserts this new user into the Users table using SQLite
  - Add new user to db screen get reloaded

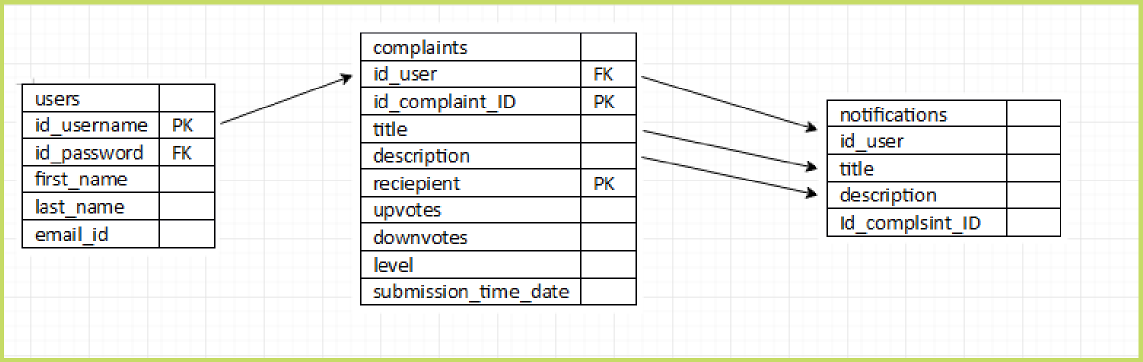

Figure 3: ERD

4 Data Storage

- This data would be stored in a database across multiple tables (in .table format) and the database would be implemented using SQLite
- The following tables are required :
- Users : stores data about all the users such as
  - firstname
  - lastname
  - email
  - username
  - entryno
  - type (0 for a student, 1 for a warden and so on)
  - password
- Complaints : stores all the data for complaints
  - Title
  - Complaint ID
  - Description
  - Level of complaint (Individual or hostel or institute level)
  - Number of upvotes

- Number of downvotes
  - Person who lodged the complaint
  - Person to whom the complaint is addressed to
  - Is resolved
  - Created at

- Notifications : stores all the notifications
  - User ID
  - Complaint title
  - Complaint ID
  - Description
  - Is seen
  - Created at

- SQLite will be used to interact between various tables (to fetch, update, insert or delete data)

- No, the database design does not allow invalid entries, this would be ensured by type-checking the fields while defining the tables.

5 API's

- The following is the list of APIs that will be used :

- Login: /default/login.json?userid=<username>password=<password>
  - POST request
  - Server returns JSON response with user details and the boolean value for success

- Logout: /default/logout.json
  - GET request
  - Server returns JSON response with the boolean value for success

- Lodge new complaint: /complaints/new.jsontitle=<title>description=<desc>scope=<scope> assignedto=<assignedto>
  - POST request
  - Server returns JSON response with the complaint details, the users details, success and the creation time
- Notifications: /default/notifications.json
  - GET request
  - Server returns JSON response with a list of notification details
- All complaints: /default/complaints.json
  - GET request
  - Server returns JSON response with a list of complaints details alongwith the users details
- Vote on a complaint: /complaints/postvote.json?complaintid=<complaintid>vote=<vote>
  - POST request
  - Server returns JSON response with the complaints details alongwith the users details and a success boolean
- Mark the complaint as resolved: /complaints/markresolved.json?complaintid=<complaintid>
  - POST request
  - Server returns JSON response with the success boolean
- Add new user to db: /default/adduser.json username=<username>password=<pass> firstname=<firstname>lastname=<lastname>entryno=<entryno>type=<type>
  - POST request
  - Server returns JSON response with the list of all users, the added user and the success boolean

6 Modularity

- The project can be divided into the following modules:
  - Server code
  - Database code
  - Android Java files for activities and other auxiliary functions
  - User interface files
  - Networking code
  - Session management and cookies
  - Google cloud messaging setup
  - Retry policy
  - Android client

7 Bug foresight

- These bugs might be encountered in the project:
  - SQLite exceptions
  - Heisenbergs
  - Gradle errors
  - Web2Py HTTP-500 (internal server) errors
  - Missing packages
  - Python CGI errors
  - General and crucial interface errors

References

[1] Tutorial on setting up server: *https://www.digitalocean.com/community/tutorials/how-to-use-the-web2py-framework-to-quickly-build-your-python-app*

[2] Design doc tips: *http://www.cse.iitd.ernet.in/cs5110300/SessionAssignment2*