Institute Level Complaint Management System

In this assignment, you have to design and develop a complaint management system for an institute like IIT Delhi. The team that gets the highest marks in this assignment can continue working on it in the summer (and get design credits) to create a finished product that can be used at IIT.

The end-users of the system will be the people of the institute such as faculties, students, and institute employees. Using this application, the end users can submit their complaint/grievance to the concerned authorities. The application must allow for three type of complaints:

1. Individual complaint: A student wants to complain about the electricity problem in his hostel room to an electrician
2. Hostel-level complaint: Hostel residents want to complain about the bad quality of the mess food to their warden.
3. Institute-level complaint: Students across the institute want to protest against the LAN ban after 1 am

In case of individual complaints, the end user's complaint should be visible to the concerned authority only. Once the complaint is addressed, the an user should be able to mark it as resolved. Consider the example of a student whose hostel room's tube light is not working properly. The student will file a complaint which will be visible to the electrician. Once the tube light is fixed, the student should mark the issue as resolved.

In case of a public complaint (Hostel-level or Institute-level), the end user's complaint should be visible to all the people who will be affected by the complaint. For example, if there is a complaint regarding the food in Udaigiri hostel's mess, the students, and the warden of Udaigiri hostel should be allowed to view and additionally give a vote(up/down) on the complaint. In such complaints, the warden should have the right to mark the complaint as resolved. Kindly note the difference here: in case of individual complaint, the individual user was able to mark the complaint as resolved, while in case of a hostel complaint, the warden has the authority to mark the complaint as resolved.

You can specify some users as *special users*. These users have the ability to add a particular user to the database. Needless to say, at the time of data entry, the attributes of the user (needed for your application) have to be stored. All the mentioned functionalities should be present in the application. However, you can be creative with the scope and the design of the application.

Your submission must contain the code of the server, and an Android client. You can implement server in any language you want- PHP, Node, Ruby, Python etc.

Implementation of the assignment:

The scope of the assignment is big, and it is imperative that you use use good design practices to implement the assignment. We strongly advise you to first focus on the design of your application, and **NOT** start with coding. Here is a list of questions you should be able to answer before starting with the coding:

1. Scope of the assignment: What is the scope of the assignment? Do you want to extend this assignment to web clients in addition to the Android clients?
2. Data storage: How will I store the data -- databases? Which tables are needed? How will I link these tables? Is my database design allowing invalid entries?
3. API's: What API's should a server expose to the client? Learn the code of Assignment 1's server to understand how to design an API. It is important to note that well designed APIs allow you to easily extend the scope of your assignment to web client's (or even Windows phone) and not just limit it to Android clients. Make a rough API blueprint specifying what are the end-points (url with parameters), what is the request (GET, POST, PUT) and the expected server response.

4. Event flow: Consider a simple command: user wants to start an individual complaint. List all the events that will be generated as a response to this command: (a) from the user application to server (b) from server's outside interface to the concerned API (c) from server API to the database, and all the way back. Note the state that is maintained at each step of this event chain. Do this exercise for all major commands. Flow chart (https://en.wikipedia.org/wiki/Flowchart) is a popular technique to create event diagram.
5. Bug foresight: Think of all the bugs that you may encounter along each step of the event flow (described above). Design test cases to stress test each such scenario.
6. Modularity: Divide the entire work into small modules, and decide the primary interfaces between different modules. Make sure that changes to one module should not affect the working of another module.
7. Interface of the application: How should the Android interface look like?

Once you complete the brainstorming on the above mentioned issues, write them in the design document. For this assignment, the submission of the design document is March 10. The deadline for the code submission will be informed at a later stage.

You can extend the server code of Assignment 1 to implement the server for your application.

Grading:

The grading for this assignment will be done on a scale of 100. For this assignment, the breakup of the grade is:

1. Design document [20 marks]: You should submit the design of the application in the latex format. The design should at least answer the questions that have been mentioned above.
2. Basic code [30 marks]: An application that works without bugs will receive 30 marks.
3. Software practices [20 marks]
    a. Code indentation [5 marks]: A badly indented code is difficult to maintain as well as debug. Use a proper editor to write your code.
    b. Commenting [5 marks]: We expect at least one meaningful comment describing each function and each source file.
    c. Modularity [5 marks]: All the big modules of the code should be maintained in separate source files.
    d. Version control system [5 marks]: Create a repository to maintain your code. We advise you to use either of this two options: *bitbucket.org*, and *github.com*. Use this repository to exchange code between team partners, and to create multiple revisions of the code. At the time of demo, we will check the commits of all the team members, and if we found that a team member is not contributing to the submission, he/she will receive 0 marks for the assignment.
4. Design credits: This is a relative grade component. Submissions which distinguish themselves will be rewarded. Here are a few points which can be considered:

    a. Good GUI
    b. Extended the scope of the application to web clients
    c. Created a good test environment for the application

Submission:

You should submit the assignment on Moodle. The assignment should be bundled in a zip file whose name is *entryno1_entryno2_entryno3.zip*. The zip must contain three folders:

1. *src:* This should contain two directories
    a. server: This stores the server code
    b. android: This stores the Android client code
2. *apk:* This contains the apk file which will run on the mobile
3. *doc:* This contains a file main.tex that contains the design of the application. You should also include a “makefile” that builds a main.pdf file from the main.tex file.

If your submission does not adhere to the given submission format, you will receive a penalty of 10 marks.

At the time of demo, you should bring your laptop and an Android based mobile on which this application can run. We will use your Moodle submission to create an apk file and we will evaluate your assignment for this apk file.

Academic Integrity: All the submissions will be checked for plagiarism. If a student group is found to be copying code from other group or a third party source, all the concerned students (including the students who made their code available to other groups) will get ‘F’ grade and a disciplinary action will be initiated against them. You can use the code obtained from Android examples on the Internet; however, while doing so, you must clearly mention the source of the code in comments.