CS251

Tempus
Project Report

Team eNRGy

Rajat Rathi
160050015

Tummidi Nikhil
160050096

Gurparkash Singh
160050112

Contents

1 Overview

Tempus, which is the Latin for "Time" is an Android App which helps the students in effective management of time. The App basically consists of two major fragments. One is the Time-Table/Events part and the other one is the Discussion Forums part. Time-Table/Schedule is automatically generated once the user provides his details which later on keeps getting modified as more events, like quizzes and assignment deadlines appear. As for the Forums part, the user gets automatically added to the Groups of his Department and the Courses he has registered for. He can both post messages on the forum as well as add events for the groups he has been added to. This was the brief overview of our App. Further details are provided ahead.

2 Implementation

In this section, the implementations of most of the features/functions mentioned in the are explained in detail. Note that the explanations ahead are from a developer point of view. To know what these functions do and how to use them, please refer the User Documentation.

2.1 Database

We created a database on MySQL which we processed through phpMyAdmin. The PHP files required for the task are hosted on an online free web server, 000webhost.

2.2 Welcome Page

The Welcome Page displays a quote selected randomly from a local database of quotes. The animation effect used here is the Type-Writer effect. The link for github repository for it has been provided in the resources.

2.3 Log-in/Registration System

When the user enters his credentials on the Log-in Page, we check in the online database if the given LDAP exists. If it does and the password is correct, the Log-in is successful. If it doesn't, the user is taken to the Registration Page wherein we gather all his details and put it in the database.

2.4 Courses/Events/Department Representation

The table containing details of all the users ("infouser") has columns related to events and courses which contain strings. This works as follows. Each course is represented by a string of length 3 which is nothing but the index of that course in the course details ("infocourses") table. If the index is not 3 digit, we prepend 0s and make it a string of length 3. This is a very convenient programming model as when using the information of the user in our App, we just need to get the courses string corresponding to a user and we can simply parse it to obtain all information about the courses he has registered for from the courses table. The

similar model is used to manage events and departments, the only difference being that each event is represented by a length 4 string and each department by a length 2 string. Using this model, of course we make an implicit assumption on the upper bound on the number of events, courses and departments as we can't represent more than those allowed by the number of digits allotted.

2.5 Adding/Removing Courses

When the user enters the Course Code of the Course he wishes to add/remove, we find the index of that Course from the table "infocourses" and convert it into a string of length 3 as mentioned in the previous section. Then we parse the string corresponding to the user's courses and perform the desired action.

2.6 Course/Department Forum

In the database, there are 2 tables used for implementation of Forums. The first table is that of Topics ("topics"). The topics table contains all the required tables such as the user who started it, on what date/time it was started. However, the most important information it contains is "topiccat", which a 3 or 2 length string depending on if the topic is linked to a course or a department. This is used to identify the course/department to which the topic belongs. The second table is the "posts" table which contains all the posts along with the information of who posted it or when it was posted. Again, the posts table also has a column of "posttopic" which is the index (in the topics table) of the topic it is linked to. When a user opens the Department Forum, all the Topics linked with his department are loaded and displayed. Similarly, the Topics linked with a course are shown when he opens the Forum for a particular Course.
Since the courses string only contains the course he has been registered for, he will have access to the Forums of only those Forums. Thus the model developed by us is secure.

2.7 Events Handling

For this part, we have a table "events", which contains the details of all the events, such as the user who initiated it, the starting and ending times and the target group of the event. Whenever a user pushes an event it gets added to the events table and the length 4 string associated is appended to the "pendingevents" of the users belonging to that group. These events show up on the Pending Events Page of the users. When a user left swipes on an event, the string associated to that event is removed from the "pendingevents" column and is appended to the "userevents". These event show up on the My Events Page of the users.

2.8 Time-Table

3 Deliverables Promised/Delivered

3.1 Log-in/Registration System

The Log-in/Registration System has been implemented using a database in MySQL processed through phpMyAdmin. In the proposal, we promised to use SSO for the purpose. However, SSO system doesn't provide us the courses that the user has registered for, which was the primary reason to use it. So, to save time, we did not use it and made our own database on an online server instead.

3.2 User Profile System

We promised to make a system of User Accounts which he can edit anytime. We have implemented that part successfully and completely. We have provided all the required features such as Add Course, Remove Course, Change Department etc.

3.3 Courses Database

We have made an online database of all the courses along with their timings and venues which upon requesting can be displayed through the appropriate interface provided in the app. The courses registered by a user has also been managed exclusively.

3.4 Forum

We proposed that we will implement separate Forums for all the Courses and Departments. This feature has also been implemented successfully and works flawlessly. All the users associated with a particular group can view the contents of that group.
EXTRA FEATURES: Earlier we thought of making the Forum just like WhatsApp groups, i.e. only one layer per course. However, we added an extra layer, i.e. all the posts are under separate threads now. So, on opening a Course Forum, a user will see all the topics, and then he can go to individual Topics to view Posts under that.

3.5 Events Management

This is a feature which we have implemented completely, but we have changed the implementation heavily from the one planned during Proposal. The current implementation is convenient for both user as well as developer. First we thought of developing a criteria of upvotes and downvotes on an event to push them into events database of each user. The new system that we have implemented is that whenever someone pushes an event for a group, that event will show up as "Pending" for all the users'. It would be up to the user to accept or decline the event. This will prevent unnecessary events to be added in a user's time-table. Also, it will prevent unnecessary spamming.

3.6 Time-Table Generation

Earlier, we thought of using the Google Calender API to display Time-Table, but we found a more convenient to implement system on github which we finally used. The link for the repository has been provided in the Resources file. There are a couple of Bugs in this part, which have been mentioned in the Bugs Section.

3.7 Alarms/Notifications

This is the only part of the App that we were not able to implement. We had planned to set Alarms and Notifications for all the events that had been added to a user's time-table. Also, we planned to notify the user whenever someone posts something on the forum. Doing all this was seemingly difficult and not possible due to shortage of time. We had to figure out a way to keep the app running in the background and show its alerts in that state. We were not able to implement this in time. Also, the problem was that we would have to always keep refreshing the database to check for changes which would be extremely inefficient.

4 Future Scope

4.1 Alarms/Notifications

This is one of the features that we promised but couldn't deliver. This can be implemented if one figures out how to show notifications/alerts related to the App even when the App is not running which we couldn't do because of shortage of time. If implemented, this feature would be really helpful for the users.

4.2 SSO based Log-in

As mentioned in the previous section, we did not implement SSO based Log-in system. If implemented, this would make the App more secure and robust.

4.3 Number of Events/Courses/Departments

To make the implementation more efficient, we have kept the length of representative strings for Departments, Courses and Events as 2,3 and 4 respectively. This means that there is an upper limit of 10, 100 and 1000 respectively. For a more widespread version of the App, the string lengths can be increased to cater a wider audience.

5 Tools/Languages Used

1. Java
2. XML
3. Android Studio

4. Android Debugging Tools
5. 000webhost
6. PHP
7. MySQL
8. phpMyAdmin
9. LaTeX
10. JavaDoc
11. Doxygen
12. Git

6 Bugs

All the bugs that are related to the front-end part have been mentioned int the User Documentation.
Please refer that for the front-end related Bugs. Following are back-end related shortcomings.

1. As mentioned before, we have represented the Events/Courses/Departments as strings of limited length. So there is an upper limit on the number of items that can be represented. So in order to use the app for a wider audience the database model would have to be slightly changed.
2. In the time-table section, the user is not able to add and delete the events which we had mentioned in the Project Proposal. This is primarily because we could not use the Google Calendar API. So currently, the user is only able to view his time table and not make changes to it.