Void Bowels - Vinay Khemlani, Hershey Harpaldas, Greg Ghahramani, Kyle Burkey

Project Documentation

BU-tiful Schedules is the future of Boston University's class registration process. The application is designed to allow users to find their perfect schedule based on the courses they are planning to take in the coming semester. It speeds up the process by displaying all possible schedules for a given combination of courses in the form of a graph and then allows users to filter out schedules based on certain criteria. Upon choosing a schedule, the application displays more information about each class including the section number used to register, professor names and class locations.

Design and UI

BU-tiful Schedules' icon is symbolic of Boston University's registration process. The smaller rectangles are representative of the rectangles used in the graphical view of schedules on the Student Link and feature colors similar to those used online. The blocks come together to form the letters "BU" and are places on a white background with a long shadow to give the icon a modern feel. The application itself features a color scheme centered around the red used in Boston University's branding. It also follows Android's design guidelines and features a themed version of many of Android's built in UI elements.

Functionality

MainActivity:

The application is broken down into three main activities: MainActivity, WorkingSchedulesActivity and DetailedScheduleActivity. The application opens to MainActivity and begins by asking the user to enter how many classes they are going to take. If the user enters a number greater than or equal to 10, an AlertDialog appears notifying the user that they can only enter a number up to 9. The application then takes this number from the EditText and creates a grid with that number of rows and 3 columns. The first column is where the user enters the college the class is in. These fields are AutoCompleteTextViews that provide suggestions for each of the colleges at BU. Once a third character is typed, the input is put through a validator to ensure that the user entered a valid college name. If the input is invalid, another AlertDialog appears asking the user to enter a valid name. If it is valid, the activity shifts the focus to the next EditText so the user can keep typing without having to click on the next field. The second column is a regular EditText for the department code of the class and allows for only uppercase characters to be entered. Once, a second character is entered it shifts to the third column. The third column also features an EditText but only allows for number inputs so the keyboard displays only numbers. Once three numbers have been inputted, focus shifts to the first input of the next row. Once the user hits the "GO!" button, the activity displays a dialog if all the forms haven't been completed or passes the data from the text fields into the GetCourses task.

GetCourses:

The GetCourses class extends the AsyncTask class so it can do work in the background. In order to get details of each of the classes, the application makes "get" requests to BU's websites to obtain class information. For example, for the class ENG EC 327, the app makes a get request to the URL

*https://www.bu.edu/link/bin/uiscgi_studentlink.pl/1449860101?ModuleName=univschr.pl&SearchOptionDesc=Class+Number&SearchOptionCd=S&KeySem=20164&ViewSem=Spring+2016&College=**ENG**&Dept=**EC**&Course=**327***

Any process, including this one, that needs to access the internet must be carried out in the background of the app which is why a child of AsyncTask was used here. The Jsoup library (http://jsoup.org/) was used in this part of the application to parse the HTML code and pull out appropriate information. The app then makes a new instance of the Course class and adds it to an ArrayList in the static CourseList class. While going through the code received from the get request, the app creates a new instance of the TimeBlock class for each section of the class to store the section information and what times for each class. The TimeBlock class stores the times as absolute distance from 12:00am on Sunday so a class that starts at 2:30pm on Monday would have a start value of 38.5. Once the TimeBlock is created, it is added to a HashMap where the key is the type of section (lecture, lab, discussion, etc.) and the value is an ArrayList of TimeBlocks for all the sections that fall into that type. Once all the data for all the classes has been received and processed, the application forms the cartesian product (using the Google Guava library) of all the ArrayLists of TimeBlocks so every possible schedule in the form of another ArrayList of TimeBlocks is created. It then goes through each schedule, orders them by the starting time of the TimeBlock and then checks to make sure there is no overlap in the TimeBlocks. If there is, it removes that schedule from the list since the user would not be able to register for those sections. Once the valid schedules are found, the application begins the WorkingSchedulesActivity.

WorkingSchedulesActivity:

The purpose of this activity is to display all the possible schedules for the user so he/she can learn more about whichever ones appeal to them. Therefore, the only view in the layout is a ListView and the adapter takes in the list of working schedules generated right before this activity was created (through the *getSchedules* function in the CourseList class). For each view in the ListView, the activity displays the schedule in the form of a graph which is again generated on BU's system by a manipulation of a URL:

http://www.bu.edu/uiszl_j2ee/ScheduleImage/ScheduleImageServlet?c1=**CAS+RN103+A1**&d1=**Tue**&tb1=**1400**&te1=**1530**&db1=20160119&de1=20160428&c2=**CAS+RN103+A1**&d2=**Thu**&tb2=**1400**&te2=**1530**&db2=**20160119**&de2=**20160428**...e=**8550133656**&**height**=412&width=**631**&LastActivityTime=**1449866344**

Each block of time has it's data inputted into the URL and the LastActivityTime is formed by storing the time on BU's system when retrieving class information and adding the time passed in seconds since it was stored. "e" is then generated by subracting the LastActivityTime from 10,000,000,000. The height and width are also calculated as a function of the screen size. The

activity uses this URL to download the image, crops it down to only the graph and displays it in the ListView. Once the image is downloaded the first time, it is stored as a variable for when the view is recreated by the ListView. A Toast dialog also appears to show the user the number of choices available. The user can also use this activity to filter out schedules that have 8AMs (startTime % 24 == 8), classes after 4PM (endTime % 24 >= 16), classes on Monday (startTime < 48) and classes on Friday (startTime > 120). The user would click on the filter button in the Action Bar to bring up a dialog where he/she can choose which filters to apply. A new Filter object is generated within the WorkingSchedulesAdapter class to perform the filtering and generate a new filtered list to use with the adapter. Once a user finds a schedule that interests them and they choose to click on it, the DetailedScheduleActivity is started and is passed the index of the schedule that the user has chosen.

DetailedScheduleActivity:

The DetailedScheduleActivity used the index of the schedule to get the WorkingSchedules object. It then displays the image and details using an ExpandableListView. The group view for the ListView simple has the name and number of the course and clicking on it reveals each section that applies for the given schedule. If multiple sections apply to the TimeBlock it displays all of them. The user can also save the image by pressing the save button in the Action Bar. This button downloads a larger version of the picture to save and saves it to the “Pictures” folder on the user’s internal storage. The user can also share the image by pressing the share button in the Action Bar. This button combines all the course and section names into a string and passes it into Android’s Send Intent so the user can choose what to do with it.