CSCI 5448 Object Oriented Design and analysis
Project Part 4 - MyCuOpportunities

Team - Narain Yegneswara Sharma, Ram Das Diwakaran, Ranganathan Chidambaranathan

1. What features were implemented?

- User will be able to signup and login using their colorado email address.
- User will be able to search for the jobs.
- User will be able to apply for the jobs.
- User will be able to view applied jobs.
- User will be able to Save jobs.
- User will be able to view saved jobs and apply for the saved jobs.
- User will be able to post opportunities.
- User will be able to view posted jobs.
- User will be able to view his profile.

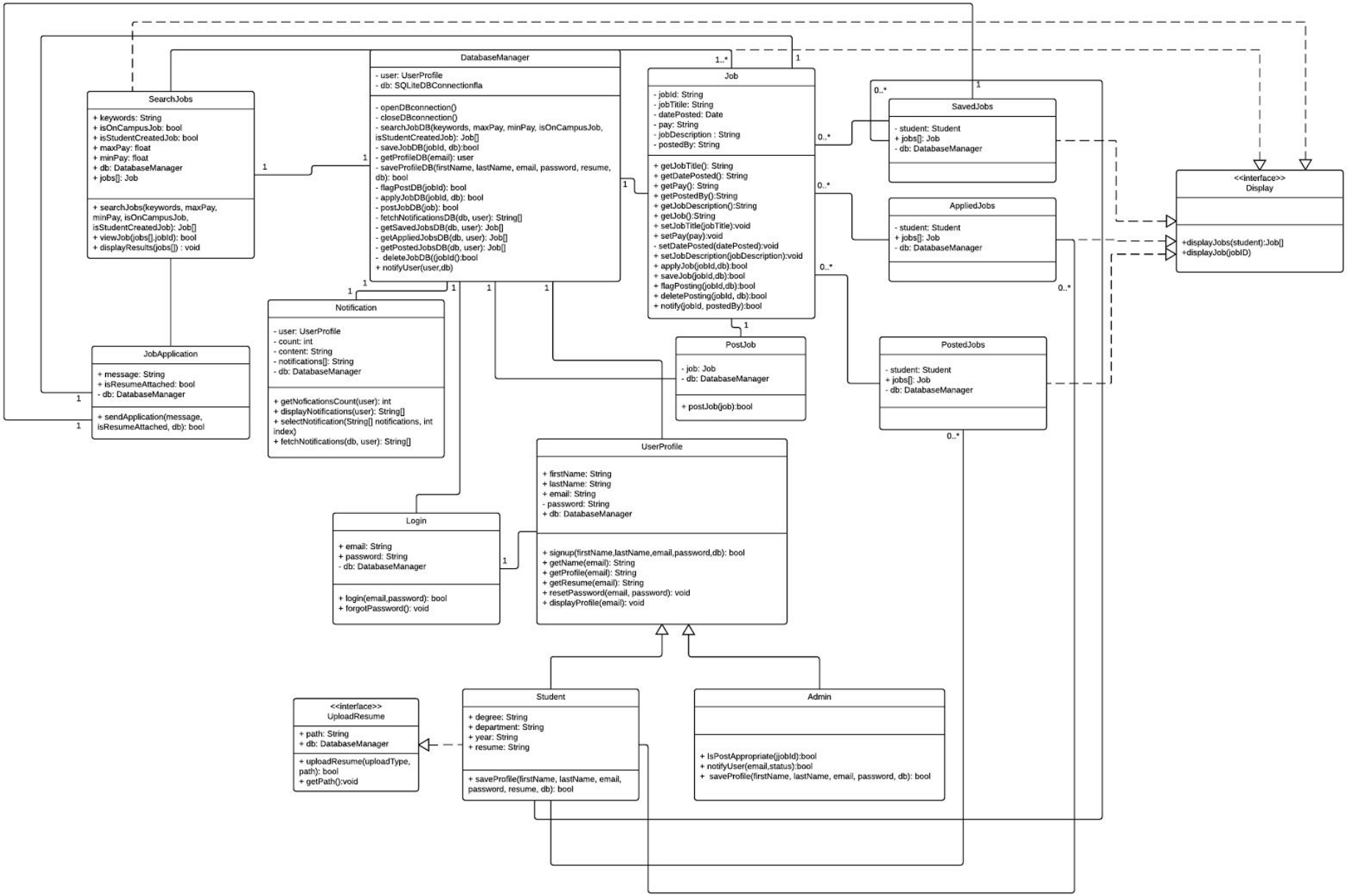
2. Which features were not implemented from Part 2?

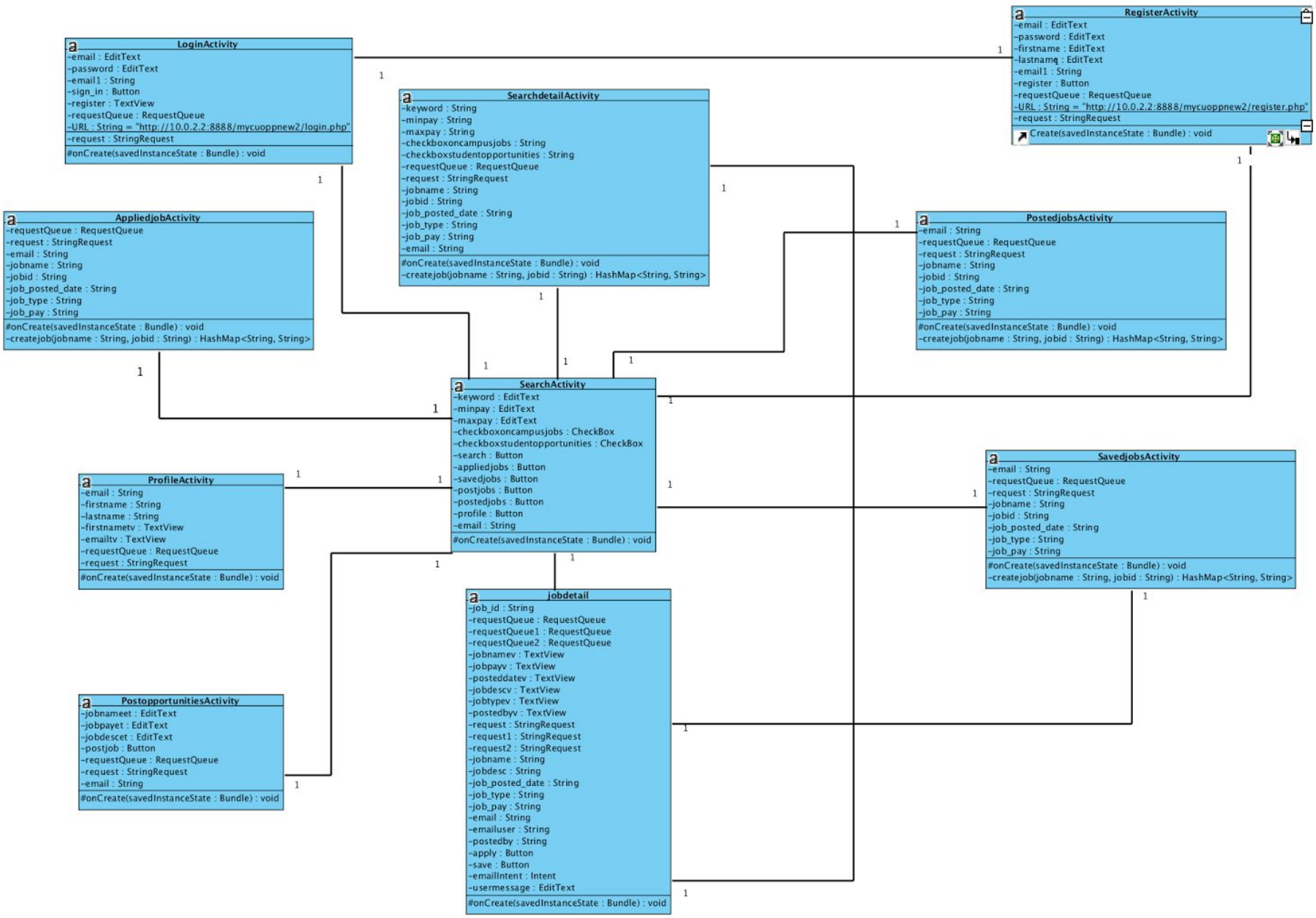
- Flagging the post.
- Upload the resume from the application. (The resume attachment option was given in the email client)
- In-app Notification / Alerts.

These features could not be implemented primarily due to time constraints.

3. Show your Part 2 class diagram and your final class diagram. What changed? Why? If it did not change much, then discuss how doing the design up front helped in the development.

The old class diagram (from project part 2) and the final class diagram are given in the subsequent pages respectively. The description about the changes are in the page that follows the diagrams.


DatabaseManager
- user: UserProfile
- db: SQLiteDBConnectionfla
- openDBconnection()
- closeDBconnection()
- searchJobDB(keywords, maxPay, minPay, isOnCampusJob, isStudentCreatedJob): Job[]
- saveJobDB(jobId, db):bool
- getProfileDB(email): user
- saveProfileDB(firstName, lastName, email, password, resume, db): bool
- flagPostDB(jobId): bool
- applyJobDB(jobId, db): bool
- postJobDB(job): bool
- fetchNotificationsDB(db, user): String[]
- getSavedJobsDB(db, user): Job[]
- getAppliedJobsDB(db, user): Job[]
- getPostedJobsDB(db, user): Job[]
- deleteJobDB((jobId():bool
+ notifyUser(user,db)
SearchJobs
+ keywords: String
+ isOnCampusJob: bool
+ isStudentCreatedJob: bool
+ maxPay: float
+ minPay: float
+ db: DatabaseManager
+ jobs[]: Job
+ searchJobs(keywords, maxPay, minPay, isOnCampusJob, isStudentCreatedJob): Job[]
+ viewJob(jobs[],jobId): bool
+ displayResults(jobs[]) : void
Job
- jobId: String
- jobTitile: String
- datePosted: Date
- pay: String
- jobDescription : String
- postedBy: String
+ getJobTitle(): String
+ getDatePosted(): String
+ getPay(): String
+ getPostedBy():String
+ getJobDescription():String
+ getJob():String
+ setJobTitle(jobTitle):void
+ setPay(pay):void
- setDatePosted(datePosted):void
+ setJobDescription(jobDescription):void
+ applyJob(jobId,db):bool
+ saveJob(jobId,db):bool
+ flagPosting(jobId,db):bool
+ deletePosting(jobId, db):bool
+ notify(jobId, postedBy):bool
SavedJobs
- student: Student
+ jobs[]: Job
- db: DatabaseManager
AppliedJobs
- student: Student
+ jobs[]: Job
- db: DatabaseManager
PostedJobs
- student: Student
+ jobs[]: Job
- db: DatabaseManager
<<interface>>
Display
+displayJobs(student):Job[]
+displayJob(jobID)
Notification
- user: UserProfile
- count: int
- content: String
- notifications[]: String
- db: DatabaseManager
+ getNoficationsCount(user): int
+ displayNotifications(user): String[]
+ selectNotification(String[] notifications, int index)
+ fetchNotifications(db, user): String[]
JobApplication
+ message: String
+ isResumeAttached: bool
- db: DatabaseManager
+ sendApplication(message, isResumeAttached, db): bool
PostJob
- job: Job
- db: DatabaseManager
+ postJob(job):bool
UserProfile
+ firstName: String
+ lastName: String
+ email: String
- password: String
+ db: DatabaseManager
+ signup(firstName,lastName,email,password,db): bool
+ getName(email): String
+ getProfile(email): String
+ getResume(email): String
+ resetPassword(email, password): void
+ displayProfile(email): void
Login
+ email: String
+ password: String
- db: DatabaseManager
+ login(email,password): bool
+ forgotPassword(): void
<<interface>>
UploadResume
+ path: String
+ db: DatabaseManager
+ uploadResume(uploadType, path): bool
+ getPath():void
Student
+ degree: String
+ department: String
+ year: String
+ resume: String
+ saveProfile(firstName, lastName, email, password, resume, db): bool
Admin
+ IsPostAppropriate(jjobId):bool
+ notifyUser(email,status):bool
+ saveProfile(firstName, lastName, email, password, db): bool


LoginActivity
-email : EditText
-password : EditText
-email1 : String
-sign_in : Button
-register : TextView
-requestQueue : RequestQueue
-URL : String = "http://10.0.2.2:8888/mycuoppnew2/login.php"
-request : StringRequest
#onCreate(savedInstanceState : Bundle) : void
RegisterActivity
-email : EditText
-password : EditText
-firstname : EditText
-lastnamq : EditText
-email1 : String
-register : Button
-requestQueue : RequestQueue
-URL : String = "http://10.0.2.2:8888/mycuoppnew2/register.php"
-request : StringRequest
Create(savedInstanceState : Bundle) : void
SearchdetailActivity
-keyword : String
-minpay : String
-maxpay : String
-checkboxoncampusjobs : String
-checkboxstudentopportunities : String
-requestQueue : RequestQueue
-request : StringRequest
-jobname : String
-jobid : String
-job_posted_date : String
-job_type : String
-job_pay : String
-email : String
#onCreate(savedInstanceState : Bundle) : void
-createjob(jobname : String, jobid : String) : HashMap<String, String>
AppliedjobActivity
-requestQueue : RequestQueue
-request : StringRequest
-email : String
-jobname : String
-jobid : String
-job_posted_date : String
-job_type : String
-job_pay : String
#onCreate(savedInstanceState : Bundle) : void
-createjob(jobname : String, jobid : String) : HashMap<String, String>
PostedjobsActivity
-email : String
-requestQueue : RequestQueue
-request : StringRequest
-jobname : String
-jobid : String
-job_posted_date : String
-job_type : String
-job_pay : String
#onCreate(savedInstanceState : Bundle) : void
-createjob(jobname : String, jobid : String) : HashMap<String, String>
SearchActivity
-keyword : EditText
-minpay : EditText
-maxpay : EditText
-checkboxoncampusjobs : CheckBox
-checkboxstudentopportunities : CheckBox
-search : Button
-appliedjobs : Button
-savedjobs : Button
-postjobs : Button
-postedjobs : Button
-profile : Button
-email : String
#onCreate(savedInstanceState : Bundle) : void
ProfileActivity
-email : String
-firstname : String
-lastname : String
-firstnametv : TextView
-emailtv : TextView
-requestQueue : RequestQueue
-request : StringRequest
#onCreate(savedInstanceState : Bundle) : void
SavedjobsActivity
-email : String
-requestQueue : RequestQueue
-request : StringRequest
-jobname : String
-jobid : String
-job_posted_date : String
-job_type : String
-job_pay : String
#onCreate(savedInstanceState : Bundle) : void
-createjob(jobname : String, jobid : String) : HashMap<String, String>
jobdetail
-job_id : String
-requestQueue : RequestQueue
-requestQueue1 : RequestQueue
-requestQueue2 : RequestQueue
-jobnamev : TextView
-jobpayv : TextView
-posteddatev : TextView
-jobdescv : TextView
-jobtypev : TextView
-postedbyv : TextView
-request : StringRequest
-request1 : StringRequest
-request2 : StringRequest
-jobname : String
-jobdesc : String
-job_posted_date : String
-job_type : String
-job_pay : String
-email : String
-emailuser : String
-postedby : String
-apply : Button
-save : Button
-emailIntent : Intent
-usermessage : EditText
#onCreate(savedInstanceState : Bundle) : void
PostopportunitiesActivity
-jobnameet : EditText
-jobpayet : EditText
-jobdescet : EditText
-postjob : Button
-requestQueue : RequestQueue
-request : StringRequest
-email : String
#onCreate(savedInstanceState : Bundle) : void

Changes to the class diagram:

We did change the class diagram from how we had envisioned it during the design phase. Apart from the change of class and method names, which were done to provide more clarity to the task being performed and to conform to a particular naming convention, we had to make a few changes due to the change in our platform and the three of us being new to Android. Initially, the plan was to use SQLite database as our datastore. In the end, we went with a MySQL database and performed the database operations through PHP scripts. This was done to offload the maximum work to the server side and limit the work being done at the Client side. We also used Volley, which is a HTTP library for Android that makes the task of networking easier and simple. We created RequestQueues for each activity on the client side which was then passed to the server to get the results.

So the major reason for changes to the class diagram could be attributed to the changes in our platform and our being new to Android. We would have loved to get the first version of the class diagram correct but being new to Design Patterns and UML, we made some unfortunate mistakes that gave us a good learning experience that we would certainly use in the future. In the future, we would look upon this experience and know better about where the errors might exist and also account for changes that may happen later on. We have learned that if done correctly, the class diagram would really save us a lot of time and trouble later on.

4. Did you make use of any design patterns in the implementation of your final prototype? If so, how? If not, where could you make use of design patterns in your system?

We implemented the following architectural and design patterns.

1)Model - View - Presenter

Case 1:

Model - PHP Script
Presenter - Search Activity(Fetches the searched job details from the model as JSON and updates the List View)
View - List View

Model - PHP Script
Presenter - Applied Job Activity(Fetches the applied job details from the model as JSON and updates the List View)
View - List View

2)Facade:

An interface that provides limited features of a bigger set of functionalities.

A Student can post/create only student opportunities but cannot post/create On-campus job. This is achieved using the Facade Design pattern.

A design pattern that we were looking forward to implement, but couldn't, primarily due to time constraints was the Observer Pattern for the in-app notifications.

5. What have you learned about the process of analysis and design now that you have stepped through the process to create, design and implement a system?

The experience of doing in-depth analysis and design before implementing the system was truly exciting. One of the key lessons that we learned by doing the analysis of the system was to get a clear vision of our system which was done by defining the requirements and ensuring none of the requirements were contradictory to each other. Drawing the UML diagrams gave us a proper framework of our system, which in turn gave us a solid understanding of the system. Drawing the class diagram and understanding how the classes interact with each other via sequence diagrams simplified our coding task later on, even though we were new to Android. We realized that if we spend quality time on the design and analysis phase it really simplifies our task later on while doing the coding. With a clear vision of the entire system in front of us, we are less likely to write code that is not modular. Getting to know about Design Patterns helped us a lot in the sense we were able to look at our use cases and identify scenarios where we could use a design phase that is well documented and would be easier to modify in the future instead of writing some custom logic that may not scale well in the future. Doing the UI design was a blessing in disguise as it gave us the exact layout of our system and we stuck to the initial UI design that we created. The final system UI came out exactly as we had planned.

We did make a few mistakes while doing the design phase, as we did not account for a platform change later on. But we are happy that we learnt a lot from our mistakes and we are sure that these lessons will help us to design a better system in the future. Apart from this, we also got a valuable lesson about the SDLC and the importance of doing design upfront.