2019

Campus Planner

TEAM NIWWD

JEFF FOERTSCH, ALEC GUILIN, CARLOS HERNANDEZ, JOSEPH SANTIBOUT

Jeff Foertsch, Carlos Hernandez, Alec Guilin, Joseph Santibout
CS 441/Dr. Zhang
Project Documentation

1: Introduction

For our software engineering term project we chose to build a personal planner for students with light social and sharing elements as an application for the Android Operating System. Using the principles of sofware engineering, as a team we determined and documented the requirements of our system, then later determined and document our system's architecture. Once we had determined these aspects of the project we implemented them, developing our project in the Android Studio IDE, using the Kotlin programming language and Google's Firebase as a database/backend host. Our UI was built in XML that integrated with our Kotlin code. During our development we used Git as our method of version control via the GitHub web service and the Git dashboard program.

Our project was a very positive and informative learning experience for all of us. During the process we learned a lot about several software engineering principles and gained a lot of experience using new platforms and languages! We generally found developing on new platforms in a new language to be the most difficult part of the software development process. Though in the end we were unable to implement several of our project requirements due to difficulties with the languages and platforms we chose, and issues with project management on the part of the project lead, we overall had a good experience developing our project.

Details beyond the scope of this report such as scope and specific requirements can be found in the requirements documentation or architecture documentation submitted earlier in this semester, or by request.

2: System Functions

The goal of the Software is to allow users to MANAGE, VIEW, and SHARE their tasks for their current academic courses:

- Users will be able to CREATE AND MANAGE AN ACCOUNT (2.2.1). Users will be able to CREATE AND MANAGE COURSES. Users will be able to CREATE AND MANAGE TASKS FOR A CORRESPONDING COURSE. The Software will PROVIDE AN OVERVIEW OF COURSES AND EVENTS VIA A PERSONAL PLANNER PAGE. Users will be able to BROWSE AND CONNECT WITH OTHER USERS VIA A SOCIAL SEARCH FEATURE. Users will be able to SHARE COURSES AND TASKS WITH CONNECTED USERS.
- The Software will contain AN ACCOUNT SYSTEM INTERFACE. The Software will contain a MONTHLY PLANNER INTERFACE. The Software will contain a SOCIAL SEARCH INTERFACE. The Software will contain a FRIEND TASK AND COURSE SHARING INTERFACE. The Software will contain a FRIEND LIST INTERFACE. The Software will contain a COURSE CREATION INTERFACE. The Software will contain a COURSE MANIPULATION INTERFACE. The Software will contain a TASK CREATION INTERFACE. The Software will contain a TASK MANIPULATION INTERFACE.
- The Software will provide these interfaces via an ANDROID OS COMPATIBLE APPLICATION. The Software will include a SERVER-SIDE DATABASE SYSTEM TO CONTAIN PROFILE INFORMATION, AND USER COURSE AND TASK INFORMATION.

2.1 - Create and Manage an Account:

2.2.1.1 - Create a New User Account:

Users will be able to create a profile account using the Account System Interface (3.1.1.1).

2.2.1.2 - Manage a User Account:

Users will be able to manage their account information fields using the Account System Interface (3.1.1.1).

2.2.1.3 – Login to User Account

Users will be able to log into User accounts using the Account System Interface (3.1.1.1). Users will be able to create a new User account using the New Account Creation Interface (3.1.1.1.2).

2.2.1.4 - Account Information Fields:

- First Name
- Last Name
- User Name
- Password
- Email Address
- Date of Birth
- Academic Institution (School) Name
- Current Year/Grade in School

2.2 - Create and Manage Courses:
Users will be able to create new courses using the Course Creation Interface (3.1.1.6). Users will be able to manipulate their created courses using the Course Manipulation Interface (3.1.1.7).

2.3 - Create and Manage Tasks for Corresponding Courses:
Users will be able to create new tasks using the Task Creation Interface (3.1.1.8)

2.4 - Provide an Overview of Courses and Events Via a Personal Planner Page:
Users will be able to view an overview of their courses and events through several formats using the Monthly Planner Interface (3.1.1.2).

2.5 - Browse and Connect with Other Users Via a Social Search Feature:
Users will be able to interact with other users via the Social Search Interface (3.1.1.3), and the Friend List Interface (3.1.1.5).

2.6 - Share Courses and Tasks with Connected Users:
Users will be able to share courses and tasks with friends using the Friend and Task Course Sharing Interface (3.1.1.4).

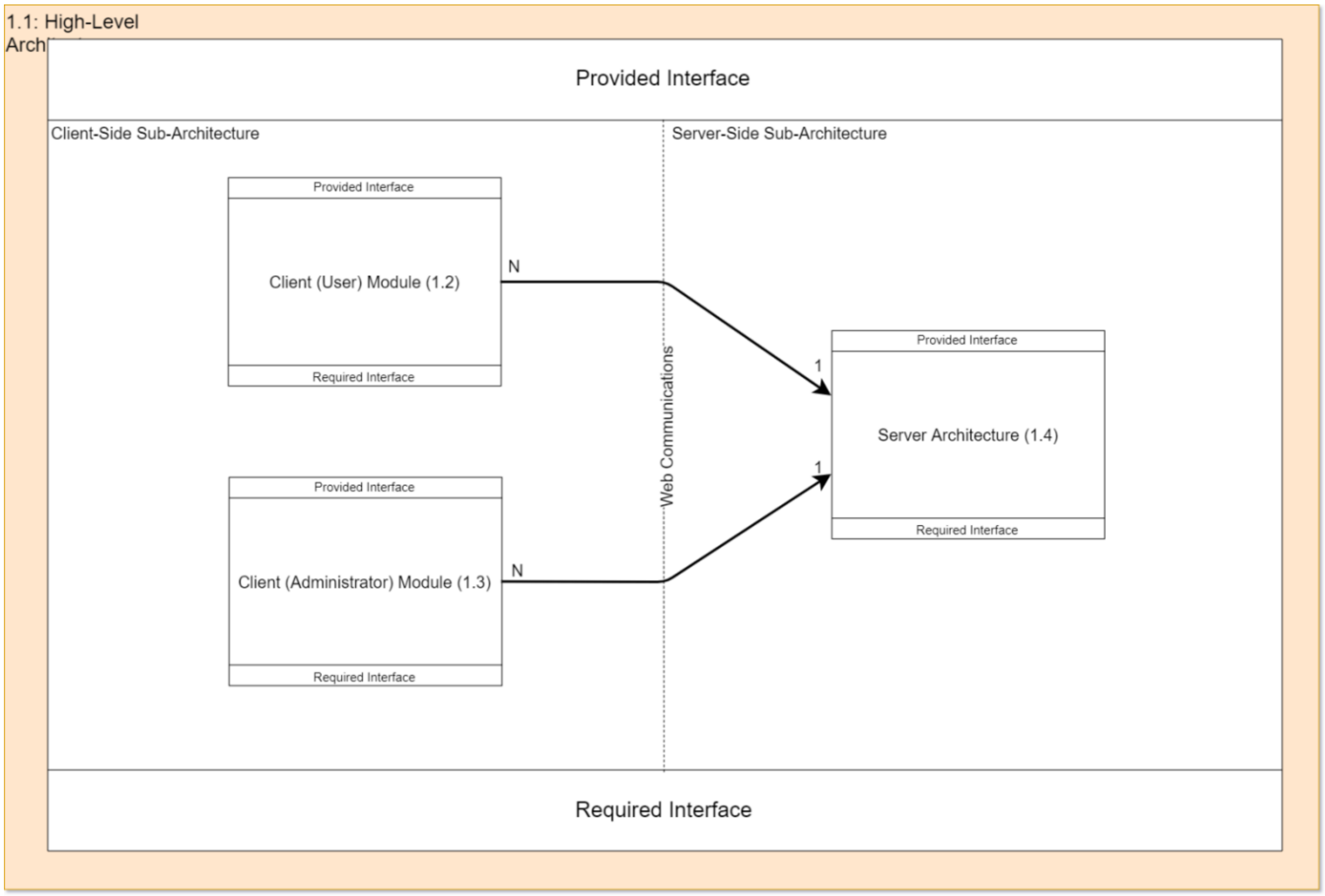
2.7 – Server-Side Database System:
This entity relationship diagram has not been included as it falls outside the scope of the current assignment.

2.8 - Android OS Compatible Application:
The interfaces and functionality of the Software will be delivered via an interactive Android OS Compatible Application (3.2.7.1).

3: Architecture

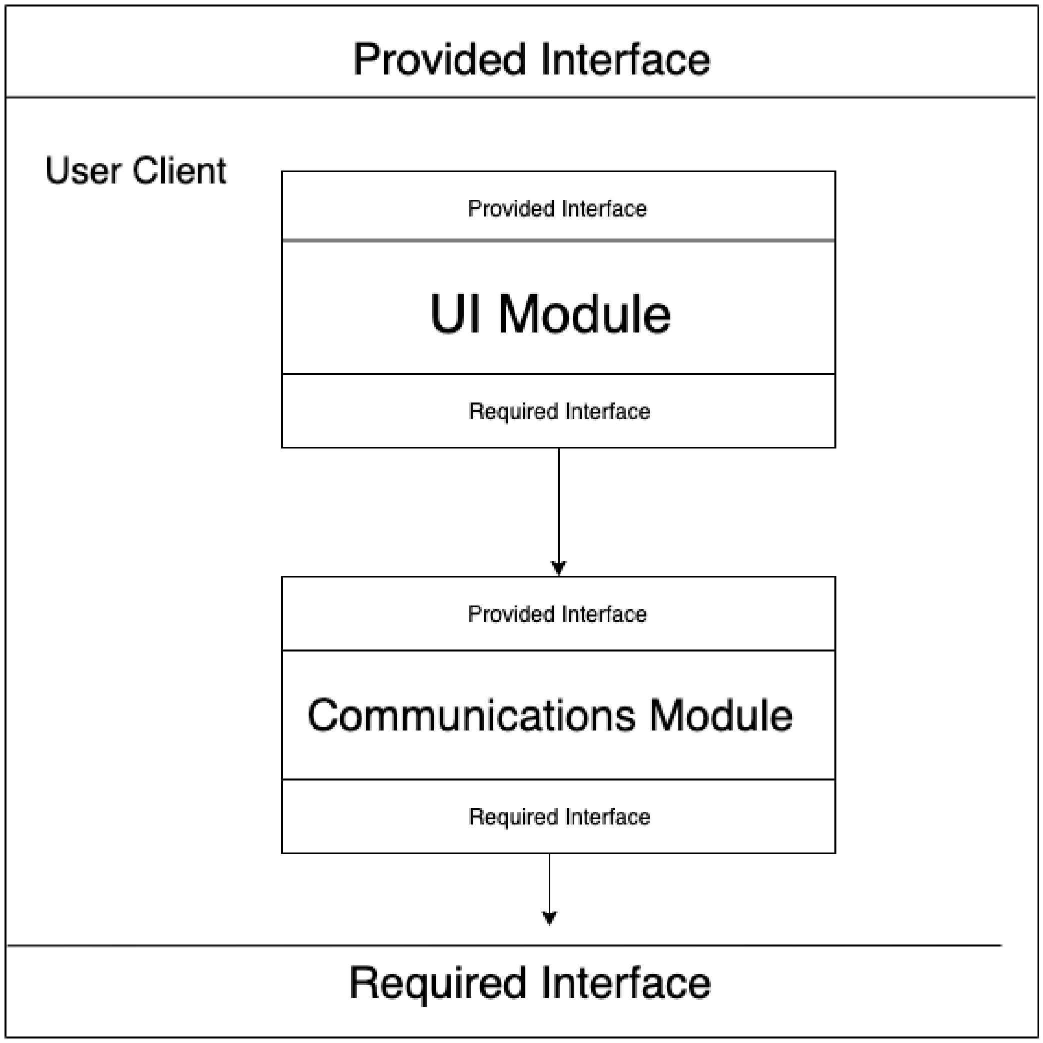
3.1: High Level Architecture Overview:

Though our high-level architecture mimics MVC style architecture in its implementation, ultimately the overall high level architecture matches a Blackboard style. This is due to the open and social aspect of the blackboard portion of our architecture, where students are able to see database information not related to their individual contributions to the system, while also contributing their own portion to the blackboard (server architectur



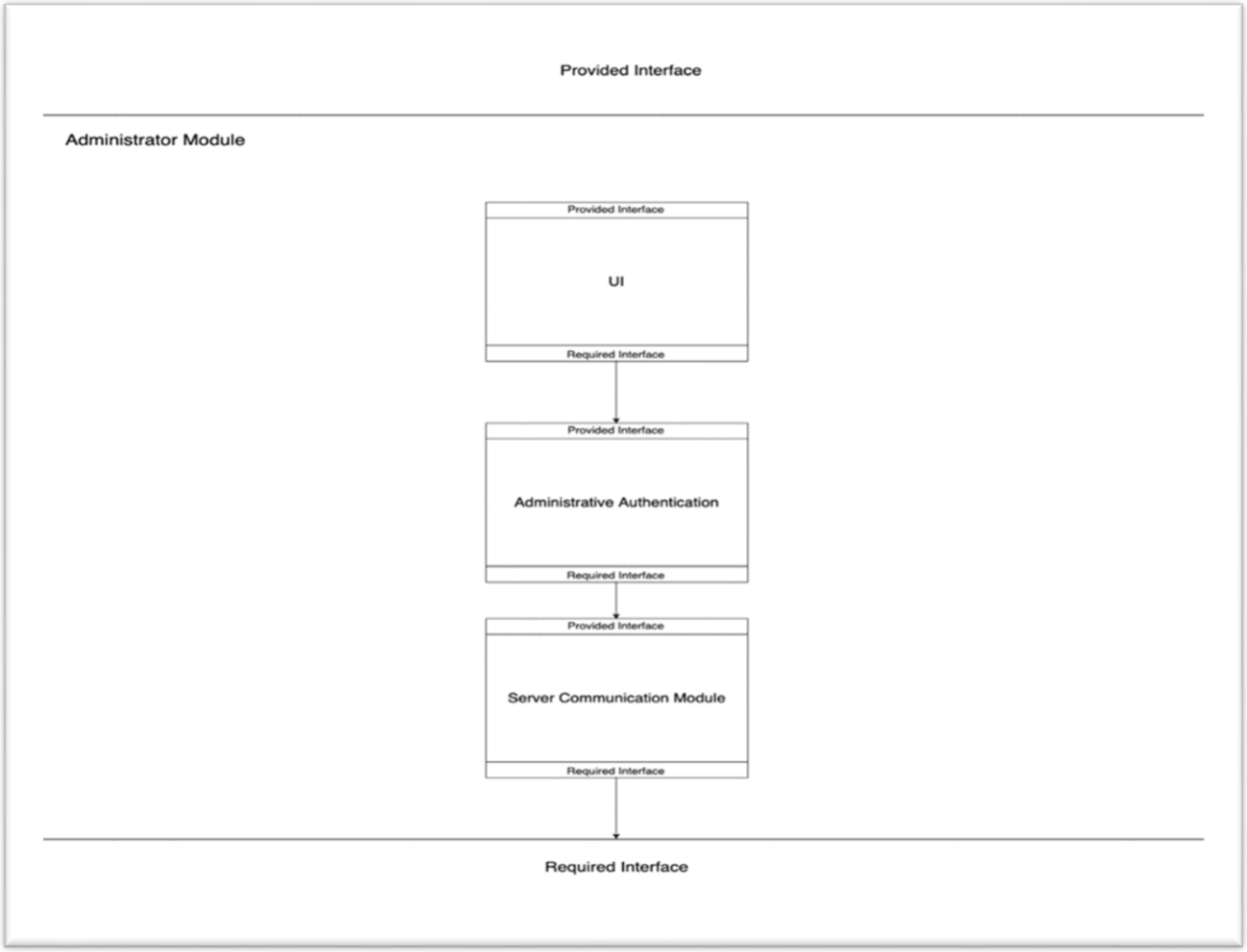
3.2 – Client (USER) Overview:

The Client (USER) module contains all sub-modules and interfaces required for the User to interface with the program and communicate with information stored on the server. More information on this module and its sub-modules can be found in section (2) of this document.



3.3 – Client (ADMINSTRATIVE) Overview:

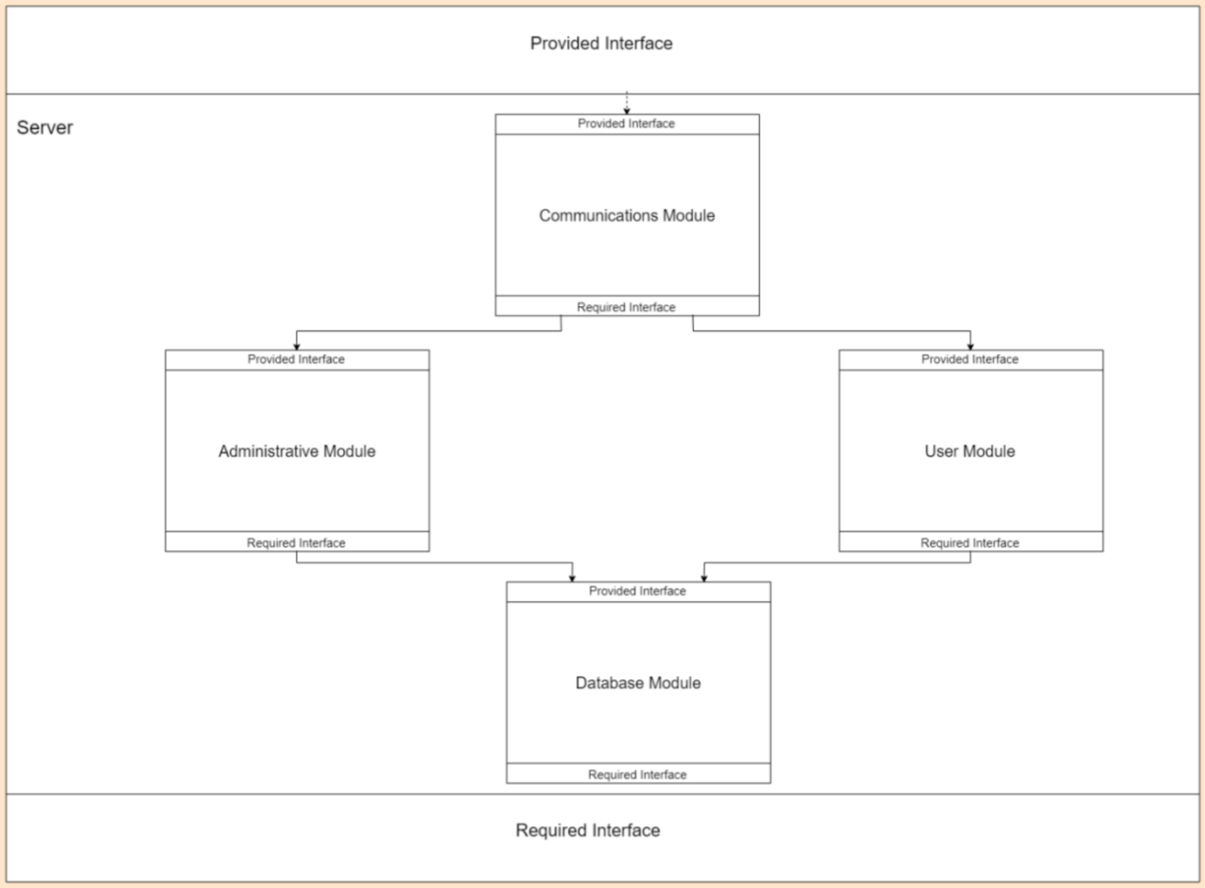
The Client (ADMINISTRATOR) module contains all sub-modules and interfaces required for the Administrator to interface with the program. The module also contains all sub-modules required to communicate with the server, and monitor its status, operations, and Client-side interactions. More information on this module and its sub-modules can be found in section (3) of this document.



3.4 – Server Overview:

The Server module contains all sub-modules required to manage and manipulate the data and services that comprise the Software. This module also contains sub-modules that handle communications between each client module. More information on this module and its sub-modules can be found in section (4) of this document.

Figure 4.1.3 - Server Module Specification



4: Implementation

4.1: Implementation Technologies

We required several technologies to implement our project, including a development environment in which to write our program, an adaptable language to write the program in, and a service to host our backend and database. An additional requirement for our development was for the technologies we chose to have enough synergy to easily adapt and integrate the components of the system. Since the platform we were developing for was the Android Operating System, we chose the default (and android supported!) IDE, Android Studio (AS). The language we chose was informed by our development platform choice as well. AS's front end is built in XML by default, and integrates with two immediately supported languages, Java and Kotlin. Though Java may have been the better (and more familiar for 50% of our project team) option, we chose to use Kotlin as our default language due to a number of factors. The documentation we researched on the language led us to believe that it was a simple, powerful language with many benefits for Android development. We discovered too late into our actual development process to change that this was not the case. Google's Firebase platform was chosen as the hosting location for our database, again for its apparent ease of use. Though we found that the actual database side implementation tasks such as creating and hosting the database were extremely simple, connecting to the database from our Kotlin code proved extremely difficult.

4.2: Implementation Tasks

The following generalized tasks were required to implement our project (not including testing):

1. Establish the Git-based version control system for our project.
2. Establish a database platform on Firebase
3. Build the front-end elements to be used by the app (the UI).
4. Build the Kotlin elements to be used by the app.
5. Establish UI element-level connections between the XML files.
6. Establish Kotlin code that connects the front-end elements of the app to Firebase.

4.3: Consistency

Though we were able to implement all of the above tasks to some extent, due to difficulties with learning these new languages and platforms, we were unable to implement many of the requirements of our project. The following list represents the requirements we were unable to implement during our development period, and explanations for why those requirements weren't implemented.

1. Manage a user account – Though the front-end elements for a profile page were implemented, difficulties with pulling information from the database ultimately led us to focus on the more critical elements of our project. If we were able to schedule our development better (this is the fault of the project lead, not the primary developers) we may have had the time needed to overcome these problems and implement this requirement.
2. Create and manage courses – This requirement was not able to be implemented in time due to reasons identical to those outlined in item 1 above. Additionally, as we continued to develop the most critical functions of our project, we determined that presenting course information in this manner was ultimately not necessary to the overall project. If we had managed our time better (again the fault of the project lead) this requirement would have been dropped, and the majority of the course functionality built into the task system.
3. Browse and Connect with Other Users Via a Social Search Feature – This requirement was partially completed. Again as in parts 1 and 2 above, the UI (XML) elements were created for this requirement, and the main friend page was implemented as part of the

final project, we were ultimately unable to implement the additional features of this requirement. We were unable to implement these for reasons identical to part 1 above.

4. Share Courses and Tasks with Connected Users – The description of this requirement's failings and their reasons are identical to part 1 above.

We were nearly unable to implement ANY of the requirements due to difficulties experienced interfacing with the database. We ultimately WERE able to interface with our database at the last minute and implement our task, login/logout, and interface requirements. Again, if our team had been able to schedule development and communicate difficulties in a better way (again the fault of the project lead), we may have been able to implement more or all of the requirements we did not ultimately achieve within the project time.

4.4: System Availability

Our system as it exists at the project conclusion (conclusion due to time, not completion) is hosted on GitHub through the following link:
https://github.com/alecguilin/CougarPlanner

5: Project Management

5.1: Role/Responsibility Chart

Team Member Name	Primary Roles and Responsibilities
Jeff Foertsch	Project Lead, Documentation, XML Layouts
Alec Guilin	Repository Management, .kt and database implementation
Carlos Hernandez	Database Management/Implementation, .kt implementation
Joseph Santibout	.kt to XML integration, XML

5.2: Project Team Problems

As a team, we worked and communicated well together. In my opinion (JF) the distribution of work was fair and allowed each team member to participate satisfactorily. Each team member contributed to the best of their ability and was willing to communicate and work together.

In terms of teamwork, I feel that we did not experience any problems. The project lead was unfortunately unable to contribute satisfactorily in that role during the ~4 final weeks due to extenuating circumstances. When this lapse in communication and contribution occurred the rest of the team was able to work well and accomplish as much as was feasible during this time. Overall it is my opinion (JF) that this was a good team and we would be willing to work together again.

6: Conclusion

Difficulties in the development process and issues outside of the course made it difficult for certain team members (project lead) to contribute adequately at certain points in the SE process caused the project to fail to meet many of its requirements. Despite this, we feel that the project itself contributed highly to our understanding of the SE process and was an overall positive experience. The lessons learned were many and varied, from principles of project management (and how to apply them) to experience with new and (in the case of Kotlin specifically) strange tools, platforms, and concepts allowed us to experience some of the realities of SE.

7: References