

*Design Document – 3rd Year Project*

*By Hugh Brady & Ryan Conway*

*Table of Contents*

*Introduction ............……………………………………………………………………………………………………………….. 1*

*System Requirements ……………………………………………………………………………………………………. 1*

*Target Audience/Market ……………………………………………………………………………………………………. 1*

*Technology Used and Why ……………………………………………………………………………………………………. 2*

*User Interface …………….…………………………………………………………………………………………………. 5*

*Database ………….……………………………………………………………………………………………………. 7*

*Implementation of the Features ………………………………………………………………………………………… 8*

*Known Bugs ………………………………………………………………………………………………… 8*

*The Future of AttendU …………………….…………………………………………………………………………….. 9*

*Conclusions …………………………………………………………………………………………………………………………… 9*

*References …………………………………………………………………………………………………………………………… 9*

*Introduction*

This software design document is to provide a description of our mobile app, AttendU, providing insight to the structure and design of the app. This design document contains descriptions and documentation of the software design for the project.

*Features*

Here are the features that we intend to include in our app;

* Log in.
* Students can take attendance.
* Students can view their grades.
* Teachers can view student attendance.
* Teachers can edit student’s grades.
* Accounts are created and edited by the administrator.

*Target Audience/Market*

This app targets schools or colleges that want an easier way to check student attendance for classes or lectures. This benefits teachers as it’s a less tedious way for them to gather attendance in classrooms. The teacher simply generates an attendance code in the app and have the students check in themselves. It also helps with the problem of students checking in other students despite them not being present as only the students who attend will have access to the code.

This app also has a grading feature that may help out with teachers handing out results. The teacher can issue out a student’s results through the app where the result is pushed to the database so the student can access that grade with ease.

*Technology Used and Why*

AttendU will be developed using Ionic 3 and will employ Firebase’s Realtime Database. Communications between the app and the database will be handled by the CRUD API.

**Ionic 3**

Ionic is an HTML5 based mobile application development framework which is used to develop hybrid mobile applications. Hybrid apps have many advantages over pure native apps, specifically platform support, speed of development and access to third party code.

We decided on using Ionic for a number of reasons;

1. Ionic allows you to create both Android and iOS devices using the same code base.
2. It has stabilized quite a bit even in the last year.
3. Ionic uses Angular, which is one of the best frameworks for modern web development, so we figured it would be beneficial to become familiar with it.

**Firebase Realtime Database**

The Firebase Realtime Database is a No-SQL cloud-hosted database. We were attracted to this database because it is compatible with Ionic and we really liked the real-time aspect of it. With some data binding you can connect your views with your data and these views will change automatically when the data changes. The performance of Firebase was another drawing point, and in our experience the performance was consistently great, although Firebase is designed with millions of users in mind, so we barely scratched the surface of what Firebase is capable of.

**CRUD**

When searching for an API to use, we considered using the RESTful API because of its robustness, but decided on using one that we hadn’t any experience with. We chose CRUD because of its prevalence and because it suited the needs of our project.

*User Interface*

**Original Design**

When we started designing the layout for AttendU, we had originally intended on using tabs to separate different features. We created a basic skeleton that we intended to use in the development of the app.

Below is a basic view of how we had originally intended the app to look.

A screenshot of a cell phone

Description generated with very high confidenceA screenshot of a cell phone

Description generated with very high confidence

**Final Design**

When we started developing the app, we felt the more fluid push and pop system worked better with the app rather than using tabs.

The design we settled on uses lists to display information and most functions have their own page.

Below is the final layout of AttendU, complete with a snazzy logo.

A screenshot of a cell phone

Description generated with very high confidence*A screenshot of a cell phone

Description generated with very high confidence*

Once the user is logged in they are brought to either the student home page, teacher home page, or admin home page depending on their account type. From here the user will be able to navigate to any page they have access to by virtue of their account.

*Database*

The database will consist of 3 tables;

1. Student
2. Teacher
3. Code
4. Attended

**Student Table**

Student table stores student information. It’s fields include;

* Email
* Password
* FirstName
* LastName
* Class
* Grades

**Teacher Table**

Teacher table stores teachers and teacher information. It’s fields include;

* Email
* Password
* FirstName
* LastName
* Subject

**Code**

Code table contains the code that will be used to take attendance. Students must enter this code to be marked for attendance. It has only one field:

* Code

**Attended**

Attended table contains the list of students that have been marked as attended by the app. It’s fields are:

* Name
* Attended

Every student and teacher will be accessed by making use of the unique identifiers that is supplied by Firebase.

*Implementation of the Features*

**Log in**

In order to get a log in functional that recognized different account types, we had to forgo Firebase’s in-built Firebase Authentication as it did not allow for us to specify whether a user was a student or a teacher. Our solution was to create lists that point to the student and teacher tables, using the subscribe and map functions to loop through the list and access the values. These values are then checked against the credentials entered by the user, and if the credentials are valid they are brought to the respective home page.

**Attendance**

In order to take attendance, the teacher must generate a code for the class. This code is then pushed to the Code table in the database and replaces any codes already present. When the student enters a code on their own app, the table is subscribed to and the values are checked against each other. If the values are the same the student is marked as attended and added to the attended list.

**Creating Accounts**

Users are created, and their details can be edited by the admin. Separate pages are used for creating accounts of different types. For the creation of accounts, a custom interface is created and a reference list points to the corresponding table in the database. Then the addStudent/addTeacher method creates the account and pushes it to the database. The account’s email and password are automatically generated but can be edited by the admin.

Known Bugs

* The user may have to press the attendance button twice to check attendance.
* The student details list displays the raw, unformatted data from the database, including some whitespaces.
* The student grades page also contains some whitespaces.
* The teacher details list displays raw, unformatted data.
* Once the admin adds/edits a student they are redirected to the root page rather than back to the list.

The Future of AttendU:

* Improvements can be made on the current app design, such as resolving the present bugs within the app at the moment. We would also hope to improve the way in which grades are issued out to students with more accurate results and better formatting.
* For further features we would like to implement a more dynamic timetable, that updates frequently and is more interactive than the current timetable. It could feature room numbers, classes and the teacher/lecturer.
* We would also look for way to possibly improve the current attendance check in system, possibly implement some date stamp to each record of attendance.

Conclusion

In conclusion, we feel that AttendU succeeded in what we sought it out to be, an attendance application. Our projected design didn’t end up the way we initially thought it would, but we are happy with the result. There were many decisions made along the way, such as choosing out database platform, application platform, design and functionality. We encountered some problems along the way, but with a little time we were able to resolve them. Overall, we are happy with the decisions and technologies we committed to and are happy without end product.

**References:**<https://stackoverflow.com/questions/40029986/how-does-map-subscribe-on-angular2-work>  
[https://www.youtube.com/channel/UCYJ9O6X1oFt7YGXpfRwrcWg/videos](https://l.facebook.com/l.php?u=https%3A%2F%2Fwww.youtube.com%2Fchannel%2FUCYJ9O6X1oFt7YGXpfRwrcWg%2Fvideos&h=ATPdeJ3bGr97ECRD9BHj3KvwzPMeicMsv28ewDbi7n0Buh_JjulsCyEmLVEO0fE2tGEd5tANioQHi3ITyRSIY0Cstk5-eeaoLfWDDf2Et6geSV23DXo-lw)  
<https://www.joshmorony.com/high-performance-list-filtering-in-ionic-2/>  
<https://github.com/angular/angularfire2/issues/848>  
<https://firebase.google.com/docs/database/web/read-and-write>