

Atomic Modules: Value Driven Design and Development of an Academic Modules Evaluation Mobile Application

Saber Hamidi, Phinephas Asare, Yanzhong Su, Ahmet Novalic, Rihards Baranovskis
{sh30g12, pa4g13, ys3n15, an2n16, rb7e15}@soton.ac.uk, University of Southampton

Abstract— There is no universal platform for finding module ratings and comments for students. Inspired by IMDB, this paper introduces idea of building one such platform. We first identify the problems from student, recruiter and university perspectives and then propose solutions, based on which we designed the architecture of our application. We used Ionic, a cross-platform framework for mobile application, built on top of AngularJS for front-end development, and node.js and mongoDB for back-end development. Datasets used in our application include module data from University of Southampton Open Data Service, map data from OpenStreetmap as well as open course data from Coursera, Udemy and Udacity. Up to date UI and UX practices were used. Business models, consisting of subscription and advertisement schemes is explored. The total cost of deploying this platform is estimated to be £250k. The potential Return of Investment is projected to reach as high a 75%.

Index Terms— open data innovations, ui-ux design, ionic, node.js, mongodb, mooc, recommender systems

I. INTRODUCTION

Most people watch movies regularly. There are hundreds of thousands of movies available in the market. Some of them are good movies. Some of them are not. Besides, different people have different preferences. They love movies for different reasons, e.g. genres, cast and directors etc. The similar thing applies to student choosing optional modules at universities. Before attending the lecture, students do not know if a certain module is really the one they would like to attend. They are struggling to find reliable comments about modules from other students. Modules are surveyed at the end of each semester. However, the feedback provided by school is usually outdated and were contributed by only a small number of students which is not general enough and cannot be served as references. We believe if there is a similar platform like IMDB where students can find ratings and comments about the modules they are going to select. It would be very helpful for them to make the most suitable choice based on their interests and preferences. Having attended a lecture about Open Data at the University of Southampton by Christopher and Dr Ash, it really inspired us to take advantages of open data. The freely available data about all the modules at the university, enabled to us see the opportunity of building a platform where students can rate and leave feedback about their experience in relation to the modules they have taken. This information can then be shared to all other students, which will help them make their mind when choosing their optional modules. Moreover,

university can also make proper adjustments based on this information which will help improve overall teaching quality and reputation.

The application also uses open data from online course providers such as Coursera, Udemy and Udacity to recommend relevant online courses for each module. This report details the structure of the application, and explains how we came up with this decision, as well as the range of technologies we used in implementing this application. The project management, future development and the business model for this platform are also discussed towards the end of this report.

II. THE PROBLEM AND THE NOVEL SOLUTION

The problem and the solution can be looked at from three angles: from student's, recruiter's and university's perspective.

A. Students

1) Problem:

As mentioned in the introduction, students have to select certain number of optional modules each semester. There is no credible source where students can find ratings and comments about the modules before they make their mind. Therefore, most students select the modules just based on module titles and module descriptions which is not informative enough.

2) Solution:

Inspired by IMDB, we aim to build a platform where students can rate and leave comments about the modules they have taken in the past. In turn, they can come back and look for ratings and comments about the modules they are going to take in the coming semesters.

B. Recruiters

1) Problem:

Nowadays, it is really challenging for recruiters to find the right students in the shortest time possible. Many students apply for a certain job. It turns out that most of them are unqualified for the job. However, recruiters still have to review their resumes which is very time consuming and error-prone. They may drop qualified students' resume by mistakes, thus losing valuable future employees. Also, recruiters regularly hold certain career events at universities to inform

students to apply for available jobs. Most of the students who attend the events might not even be their targets. Therefore, the events are usually inefficient.

2) *Solution:*

Students are the target users of our platform. They are also the target of recruiters. Therefore, our platform can serve as a bridge to connect students with recruiters. Based on students' personal interests and preferences, we could help recruiters to target students who have certain interests with high precision. For example, if student's selected modules are mostly related to machine learning, this implies that this student is likely very interested in machine learning. We could recommend this student to recruiters who are looking for students in that area.

C. *Universities*

1) *Problem:*

Universities, as places for students to study, have to enroll certain number of students not just from the U.K, but also from around the world every year. In order to fulfill this, it is vital for universities to maintain their teaching quality at a high level so that they can obtain good reputation among parents and students. One of the challenges of achieving this is that universities do not have a unified platform where they can collect feedback from students. The current survey platforms most universities have are web applications and are outdated.

2) *Solution:*

Equipped with this platform, universities are able to collect up to date feedback from students about how well the lectures were delivered. If students are not satisfied with a module, they can share their feelings on our platform. Universities can, therefore, investigate the cause of the dissatisfaction and make corresponding changes instantly. Once students' complaints are well addressed, universities would certainly gain a positive image from students. In return, students would recommend good universities to their friends and families who are potential students there. This will result in more enrollment, which in turn would make universities a better.

III. FUNCTIONAL SPECIFICATION OF THE APPLICATION

In order to realize our idea, we decided to build a mobile application supporting the mainstream mobile operating systems including iOS and Android. There are 5 main features supported by this application

A. *Module Assessment:*

In this application, we have the list of modules offered by University of Southampton. Students can come and register as a user and then search for the modules they are interested in. They then check the module rating which ranges from one star to five star (low to high quality). They can also check the comments written by previous students. Based on this information, students are able to make reasonable choices when selecting the modules. At the end of the semester,

students can come back and offer ratings and comments to the modules that they have taken. This function is the core function of our application.

B. *Rate lecturer:*

Students rate lecturers based on the quality of each lecture and anonymously provide instant feedback to lecturers.

Check lecturer's ratings:

Help students make decision while selecting modules. Lecturers know exactly how bad or well their lectures are and can make changes accordingly. Provide guidance for University to decide whether or not to hire or fire a teacher.

C. *Module wiki page:*

This feature is very useful for students. Every module would have a unique wiki page featuring the content and notes of every single lecture of that module. Every student enrolled in this module can contribute to the wiki page. Ideally, the wiki page will become the place for students to find answers and solutions to the questions and problems they can encounter.

D. *Module & Exam Timetable:*

Students can also sync their weekly timetable with our calendar. Then they can check their module timetable on the go. Besides, our application also has a map which can help to navigate students from their current location to the exact building where the lecture is going to take place.

E. *Student searching:*

This function is customized for recruiters. Based on students' preferences, we can segment them into different groups. Recruiters can then filter groups of students, which they can contact or arrange an interview with. However, due to privacy reasons, students have to explicitly provide permission for their information to be shared with recruiters.

F. *Check module marks:*

Check other students' marks (permission needed);

E.g. Student A wants to pair up with someone who has relatively higher mark in Machine Learning to do a project.

G. *Module Recommendations:*

Often, students find the content they have learned from lectures is inadequate. They usually spend a huge amount of time finding good study materials. As the content on the Internet is overwhelming and varies in terms of the quality, it is not easy to find high quality content. To solve this problem, we would recommend students with high quality lectures from Coursera, Udemy and Udacity based on their selection of the modules.

H. Other Features:

1) Social Networking

Based on students' interest group, our application would have a social networking feature that helps build connections in between students with the same interest. They can then contact each other and share their ideas towards certain problems in certain fields, thus helping improving each other.

2) Letting Service

Students may change place of living when their house contract expires. For fresher students, they need to find a place to live while studying. Our application would have a letting feature that helps connect students with landlords close the campus. Students will be able to find reliable accommodations very quickly.

IV. FRONT-END DESIGN

A. Front-end Technologies Used

Building a scalable mobile application which could run cross platform was one of our key goals, because we believed that this would increase the application's cross compatibility and therefore allow all students to benefit from Atomic Modules.

Technology Name	Pros	Cons
Ionic v2 + Angular	<ul style="list-style-type: none"> -Open source -Cross compatibility -Use common web languages (HTML, CSS, JS) -Strong community support -Write once, run everywhere -Relatively low learning curve 	<ul style="list-style-type: none"> -Performance can be questionable on older Android OS versions -Dependency on too many external libraries which when updated could lead to app instability -Application development can be slow due to the app building times
React Native	<ul style="list-style-type: none"> -Open source -Cross compatibility -Performance -Strong community support -Code changes reflected in emulator quickly -Large Variety of plugins available 	<ul style="list-style-type: none"> -Learning curve can be steep -Basic documentation in comparison to Ionic and Native development -Backward compatibility with older OS is limited -Performance tuning required
Native	<ul style="list-style-type: none"> -Performance and Stability -Strong community support -Great debugging tools -Legacy OS support -Detailed documentation 	<ul style="list-style-type: none"> -Steep learning curve, -Can only run on one ecosystem -Platform restrictions to develop an application exist e.g. "Xcode" cannot run on Windows

Table 1 - Comparison of Front-end Frameworks [1, 2, 3]

Taking into consideration the various constraints that the project faced, most notably time, we decided to use a framework that enabled us to programme the application once, yet run on Android, iOS, Windows Phone and the Web. As shown in Table 1 we narrowed the choice to 3 frameworks to create the final solution and analysed them. We decided to use the Ionic framework v2 which meant that we could develop Atomic Modules using common web languages such as HTML, CSS, JavaScript and TypeScript meaning that our learning curve wouldn't have been as steep as with other frameworks. Furthermore, the Ionic framework not only allowed us to develop the application cross platform (Windows Phone, iOS and Android) but it also meant that our application could be run as a responsive web app.

Although, it is to be noted that while both react native and native frameworks would have provided a better performance [4], we felt as though the difference gained would not have outweighed Ionic framework's benefits. Furthermore, Native or React Native learning curve could have meant that we might have missed some deadlines due to the team not having any real experience with frameworks. Also, thanks to the Ionic framework's strong community support, we were able to solve any issue encountered when developing the application with ease which helped us finish the application in time and test it thoroughly before showcasing it during the demo and presentation day. Developing Atomic modules using Native development could have been a good option, however we quickly found that this would have meant developing 2 applications for different systems, strengthening our decision to develop the application using Ionic framework.

B. User Interface and User Experience

The User Experience (UX) and User Interface (UI) are key to areas of a software development strategy that could affect the overall success of a system [5]. In order to ensure the success of Atomic modules, we aimed to provide the best user experience possible by incorporating Nielsen usability principles right into core of our design solution [6]. Firstly, we set about defining some user experience and usability goals, (see Table 2) as this ensured that our final design would meet our expectations of the way the application should have behaved.

Secondly, to create an easy to learn, effective to use and an enjoyable experience, the following guidelines from Nielsen usability principles were applied:

- **Visibility of system status:** The users should always know what the application is currently doing and be notified of any action within a reasonable time
- **Match between system and the real world:** The system should communicate with the user using a language, a concept or metaphor that the user can understand.
- **User control and freedom:** Allow users to do and undo actions

- Consistency and standards: The design language should flow so that users won't be left confused about what to do next.
- Error prevention: The system should be designed ready to prevent errors and catch unexpected errors.
- Recognition rather than recall: Make objects, actions and options visible to the user so they will be able to use the application straightaway rather than through trying to remember how they performed an action earlier.
- Flexibility and efficiency of use: An application should provide efficient and steady performance throughout its use.
- Aesthetics and minimalist design: Views should only provide information relevant to the page.
- Help & Documentation: Provide users with some instructions or help when necessary.
- Recognise, diagnose and recover from errors: Simple and meaningful error messages that users can understand and learn from should be used within a system.

Goal Type	Goal Name	Description
Usability	Effectiveness	The application should be able to achieve its intended purpose with relative ease
Usability	Efficiency	The application should not require a large amount of steps to accomplish tasks
Usability	Learnability	The application should be easy to learn and not require any detail operational manual
User Experience	Helpful	The application should be helpful to the user in achieving their objective
User Experience	Engaging	The application should be interactive and contain engaging features
User Experience	Supporting creativity	The application should allow for the user creativity to be expressed through.

Table 2. User Interface and Experience goals

As the following figures show, we developed a short application onboarding tutorial to explain some of the key features that users can take advantage of when using Atomic Modules. This is important as it will welcome new users in a friendly way thus meeting the “Help & Documentation” from Nielsen heuristic usability principles.

Furthermore, to ensure that we met our “Learning” usability goal, we decided to use figurative speech where possible to increase the user perception of the content being displayed to

them e.g. magnifying glass to search or a house to signify the home screen. Also, to further improve the overall application learnability we used a consistent design, throughout the whole application (Figure 2).

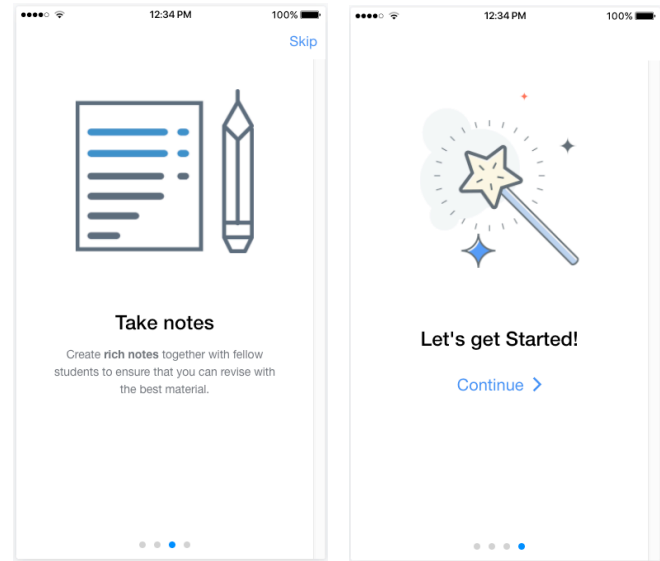


Figure 1 - Tutorial Views

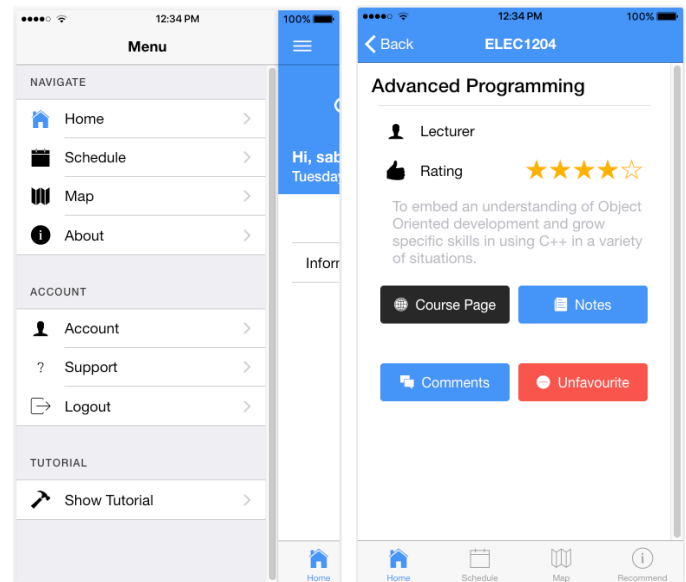


Figure 2 - Use of consistent design across views

To meet our usability goals (Efficiency and Effectiveness), we decided to create a home screen where our users would be able to accomplish the key tasks by either searching for a module using the magnifying glass icon, or by simply clicking on their favourite modules to access all the key information they may need.

Material design is a design language developed by Google [7]. Material design uses lights, shadows, material physical properties and movement to create an intuitive and simple to use user experience, implicating that when users first open the application, they will be able to use it without needing any help. Also, material design guidelines enforce the correct use

of contrasting colours to captivate users' attention to specific important actions, thus leading to a decrease in the amount of errors that users will make while using the application developed [8]. As shown in the following figure, the material design language was carefully implemented where possible into the final design our application. For instance, in the comments page, a shadow effect was added to the container to show each comment and its rating, thereby enabling comments to stand out and be easier to read.

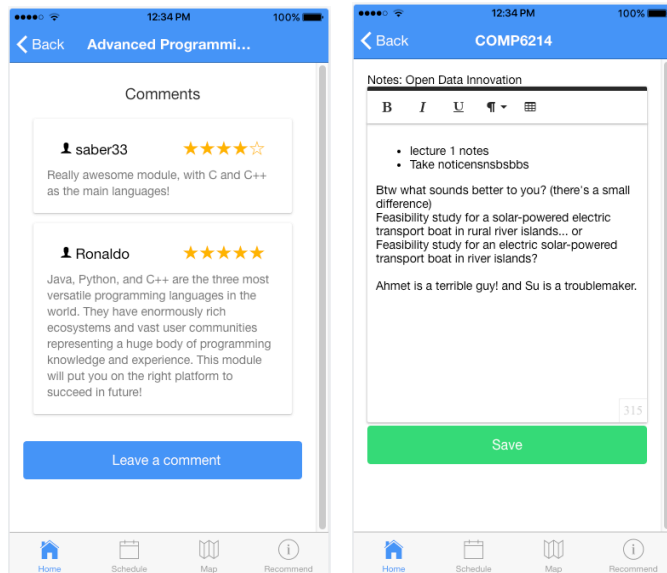


Figure 3 - Comments and Notes Views

Moreover, the colour scheme decided for the application was a Blue (Crayola) as we wanted the themes associated with the colour blue such as: trust, intelligence, authority and knowledge to be perceived by our users [9]. In addition, blue is a colour associated to social and community like feeling, meaning that its inclusion could enable the users to subconsciously engage more with the application and therefore take advantage of the collaborative note taking feature [10].

V. BACK-END DESIGN

A. Backend Technologies Choice

The mobile application requires a Backend API to implement the authentication services, as well as communication infrastructure between the cloud database and the application. There are multiple leading frameworks available for engineering a RESTful API, comparison of some is shown in the Table 3.

All three frameworks creating a RESTful, JSON based API, as well as creating direct CRUD services to maintain the database. Django and Express.js based frameworks use Python and JavaScript programming languages respectively, which both are loosely typed, dynamic languages [11] [12]. As result, they allow producing applications faster and more effectively, but the robustness of the application suffers, which

opens room for producing more development bugs. In contrast, ASP.NET core is based on C#, which is a compiled OOP, strongly typed language, which requires it to follow robust type rules and to be compiled at every change. This significantly increases the “time to deployment” and is not preferred for prototyping. Moreover, it was launched only recently to support Linux based platforms, which in our case would increase chances of running into compatibility issues when deploying our backend on the Heroku cloud application platform. All three frameworks have rich native or 3rd party features support [13]. Express.js is more lightweight compared to the rest, hence has gentler learning curve. Based on all pros and cons, Node+express.js was chosen as the tool to develop the backend logic for the application, due to easy learning curve, fast prototyping and native JavaScript support, which allowed to use JavaScript for each step of the application development.

Technology Name	Pros	Cons
Django REST framework	<ul style="list-style-type: none"> -Python based, easy to develop. -Great documentation, mature developers base. -Browsable API support 	<ul style="list-style-type: none"> -Async functionality achieved only with multithreading - Steep learning curve, as it is based on feature rich Django framework.
Node+Express.js	<ul style="list-style-type: none"> -JavaScript Everywhere, no compilation -Event driven, async framework -Lightweight, easy to start, quick to prototype -Rich NPM packages support 	<ul style="list-style-type: none"> -Harder to maintain big code bases with JavaScript -JavaScript can lead to poor coding practices -No built in errors and exceptions management system
ASP.NET Core	<ul style="list-style-type: none"> -C# forces to use good and robust coding practices -Rich in out of the box features -Built in sophisticated authentication features 	<ul style="list-style-type: none"> -C# is a compiled language, reduces development speed -Recently launched, still raw on the Linux systems -Steep learning curve

Table 3 - Comparison of Backend Frameworks [11][12][13]

Another design decision that had to be made is which database type to utilise. Both relation and non-SQL could be used for the application. MongoDB, document based, non-SQL database system was chosen due to initial lose specification of the project. Document based databases are forgiving for

changing database design in the middle of the project and exhibit great scalability and speed [14]. As our database was populated with new open data sources throughout the whole project, the loose scheme definition support was a crucial requirement. It allowed us to perform continuous database pre-population with data from different sources, without need to redesign the database at each step, which increased the development efficacy and ensured easy data transferability from one design to another. Although, due to its limited documents relations support, it did require extra effort to implement relationship among users and modules, modules and comments/recommendations. Comparison of both database types can be seen in Table 4.

Database Type	Pros	Cons
Relational, SQL based	-Takes extra care of data integrity and relationships. -Has robust, decades proven implementations	-Harder to scale and port to distributed systems. -Change in schema can result in lost data
Document, non-SQL based	-Flexible, can be changed without much effort, great for loose projects -Highly scalable and fast in performance	-Relatively new, vulnerable technologies. -Vague and schema relations implementation, migrations can lead to inconsistencies

Table 4 - Comparison of SQL based vs. Document, non-SQL based databases [14]

B. RESTful API Design

The backend API was implemented following the RESTful architecture. REST stands for Representational State Transfer and it adheres to a stateless client-server communication protocol [15]. In our case HTTP protocol was used, which allowed for implementing the necessary CRUD (Create/Read/Update/Delete) services in relation to the database management. In case of HTTP, the POST method is used to create, GET is used to read, PUT to update and DELETE method to delete [16]. The main advantages of a RESTful API are its simplicity and statelessness, which means that the backend server doesn't maintain any state or sessions regarding to the user who tries to access it. As result each restful communication request contains all the necessary information for the server to complete it, which then sends the corresponding stateless response. Other advantages of REST over Web Services as SOAP and WSDL include richer support of data formats such as JSON and CSV, better performance and scalability, and it's much wider adapted among most major web companies [17].

In total, our API was designed to have 8 different endpoints. The full list of these, with descriptions, corresponding HTTP operations and request/response JSON bodies can be seen in Appendix 1. The /signup and /auth deal with user registration and authentication correspondingly. The user authentication mechanism was architected via JSON Web Tokens [18], which encode the user information in combination with some "salt" into a JWT token and provides it back to the user. Consequently, each time user attempts to perform an authenticated action, it has to provide an Authorization header with the token in the corresponding HTTP request. Thereby server verifies the identity of the user by checking the tokens provided for the right encoded information, as result no state or a session is required to be maintained on the server side. The /modules/{:id}?fields=<field,> endpoint returns information about the specified module by its id in the database. It also accepts the fields query string, which allows the mobile application user to specify which information about the module has to be returned back. The /modules/find/{:name} implements the search functionality of the application. It utilises the regex pattern matching capabilities of the MongoDB query system and return the partly or full matching array of the modules, which match the name specified in the URL of the request. When the module is accessed by a particular user via /modules/{:id}, the corresponding feedback and rating made by the user previously is automatically inserted into the returned module's object by the server-side, in order to reduce computations required to be done on front-end mobile application, thereby improving the application's speed.

VI. OPEN DATA INTEGRATION AND THE RECOMMENDATIONS SYSTEM

One of the main goals of creating this application was to demonstrate ability to apply validation, cleaning and transformation of multiple datasets. In this project, multiple datasets were combined and reused in order to make the application experience rich and based upon up to date data.

There were three main sources of open data used in this project. First came from University of Southampton Open Data Service. Second was used from multiple MOOC (Massive Online Open Course) providers: Udemy, Udacity and Coursera. The data from these providers was consolidated and used to make the recommendation system for modules work. Last, the mapping data from OpenStreetMap and University of Southampton was used to show the map-view, with buildings and room overlays, in order to aid with navigation purposes.

A. University of Southampton Open Data

University of Southampton provides an Open Data Service available at <http://data.southampton.ac.uk>. University of Southampton, with the legacy of the Sir. Tim Berners Lee, is a

strong supporter of ‘5★ Data’¹, trying to offer datasets that are linked to other datasets, as well as making the data available online, in structured non-proprietary format, with URIs to provide connections between data [19].

The main dataset used in this project is the ‘Academic Courses’. The dataset is available in multiple formats, and conforms to 4★ Data, so it is not linking to other datasets, but it is providing all needed data about courses at single endpoint, which is very convenient and provides an easy way to integrate it in the application [20].

The data about courses is quite detailed, the RDF file contains more than 6 million rows, and provides numerous fields, such as: MAJOR_CODE, DEPT_CODE, PROG_DESC, PROG_CODE, COURSE_DESCRI, SUBJ_CODE, ACADEMIC_SESSION. In total, there is 2963 modules, most of them having complete information as outline before. Some data is missing though, for example some modules have missing lecturers, and in cases where a lecturer is present, it provides not just the name, but email, link to its open-data service page. This makes it very convenient for building rich applications that are using multiple data sources.

For convenience, data was moved to mongoDB no-SQL database and extended with other fields, like ratings and information relevant to recommendation system.

B. MOOC Sources: Udemy/Udacity/Coursera

Second major source of data are the module names and descriptions from the MOOC sources: Udemy/Udacity and Coursera. As one of the features in the application was to provide recommendation for external modules, it was needed to feed in the data from other providers in order to find the relevant modules.

Udemy, Udacity and Coursera are among the top providers of online courses² and were chosen to show recommendations from different sources. Udemy provides a more specialised approach, and according to the about.udemy.com website, their offering can be summed as ‘a global marketplace for learning and teaching online where students are mastering new skills and achieving their goals by learning from an extensive library of over 45,000 courses taught by expert instructors.’ [21]. Udacity and Coursera offer more traditional approaches, offering university-like courses and certifications with curriculum built by both industry experts and other universities. These three sources offer quite extensive ranges of courses, so all modules available at University of Southampton have multiple recommendations taken from these sources.

MOOC providers used here all provide an API access to their course databases. Udemy has a section for developers³ where

more details can be found about their course-listing APIs which were used. Udacity provides a catalogue API⁴, and Coursera has a developer program⁵.

All these sources were used together with scraping in order to get the key details about each module, which was then used to build a content based recommendation system.

C. Recommendation system

As previously mentioned, recommendation system is one of the key components of providing value to users and offering them ways to expand their knowledge by finding relevant courses. Based on the user’s current list of modules, the recommender system will offer courses from online providers that user might want to explore into more depth.

Recommendation system is built using latest machine learning technologies, by creating an affinity matrix with linear linkage. This matrix contains all of the courses from UoS courses database, as well as more than 30 thousand courses from online providers, and provides a similarity measures between those two datasets.

In order to measure the similarity and recommend relevant courses, term-frequency inverse-document-frequency (tf-idf) was used. This approach is used to see the relevance of key terms that appear in a set of documents [22]. First part of this approach, term frequency, measures how many times a term appears in the document, and ranks them according to the frequencies. Second part, inverse document frequency, is used to normalise the rankings by giving lower values to terms that are appearing too often (like ‘the’, ‘and’, ‘among’, ‘between’) but carry little value regarding the actual content of the document.

This method was combined and applied to the title, as well as the content of the modules. This resulted in a matrix that showed relationships between modules, used in the recommendation feature. As the computation of the similarity is computationally expensive, is it not viable to calculate it upon every request. Therefore, process was performed using sk-learn⁶ machine library for Python programming language, and the resulting recommendations were added to the courses database, so they are convenient to retrieve and show inside the app.

Recommendation system that is fully functional in the app shows the culmination of several tasks in the data science pipeline: collection of datasets, cleaning and transformation, combining multiple datasets together, and finally enhancing the datasets by providing new features that are used later in the application.

¹ <http://data.southampton.ac.uk/5star.html>

² <https://www.class-central.com/providers>

³ <https://www.udemy.com/developers/>

⁴ <https://s3.amazonaws.com/content.udacity-data.com/techdocs/UdacityCourseCatalogAPIDocumentation-v0.pdf>

⁵ <https://building.coursera.org/developer-program/>

⁶ <http://scikit-learn.org/stable/>

VII. SUGGESTED MONETIZATION PLAN

The aim of our application goes beyond enabling students to rate modules and leave feedbacks about their experience and overall teaching and assessment process. The future potential is to make Atomic Modules be a platform for connecting students with recruiters. Every recruitment cycle companies spend £m in pursuit of students with top skills and experience they required. The aim of Atomic Modules is to make this process less costly and less time consuming, as well as making it easy for both recruiters and students to be matched together. This is possible because the more students contribute to using this App, the more data becomes available, which then can be used to categorise students based on the modules they been performing well, and had positive experience. This information can be then fed to external companies to enable them find students with the relevant experience they are looking for. For example, if NHS needs to recruit five top graduates for the role of cyber security, Atomic Modules knows exactly which students have taken cyber security related modules and have done well, and they gets recommended to the NHS to be hired. The business model for the Atomic Modules is to monetise the access of recruiters to the student's information. This means that they will pay a monthly or yearly subscription, which will enable Atomic Modules to generate its revenue. Students on the other hand will pay with their data to be shared with external companies, and it is the data that fuel the whole process. A group of three developers are required full time to develop and deploy the App, as well as constantly maintaining and improving its functionality. Therefore, monetisation is important to generate revenue for covering the costs and keeping the Atomic Modules alive.

A. Target market

The primary target market of the system will be students and recruiters. The students will have to give their consent for sharing their academic data with the recruiters and as a result they will get hired. This way both the students and the recruiters will benefit from the system, as seen in Figure 4.

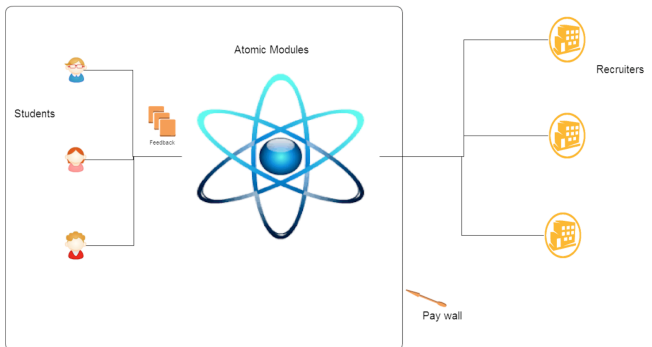


Figure 4 - High level diagram of the system user interaction.

B. Cost and Benefit Analysis

The Salary of a developer in the UK is around £45k [23], therefore hiring three developers will cost £135k/year. The advertising will also cost another extra £65k [24], and it gives

as the total cost of £200k to develop and deploy the Atomic Modules as well as maintaining and keep improving its functionality.

As of 2017 there are 2.28m students in the UK [25]. With the 20% potential users, the system will be flooded with academic data. This will attract external companies to become our customer and with an average subscription of £1/year. An estimate of 250 recruiters sign up to the system will generate a total of £250k/year, and as the number of users increase in the future years, this number will grow bigger. Given the cost and benefit figures the Return on Investment is calculated as 75%.

VIII. CONCLUSIONS

Due to the limitations of the current university modules feedback system, we have developed a novel, cross-platform mobile application. This application provides a way for students to make objective judgments about other students' experiences, which have taken their modules of interest. As result, this application creates a unique platform for the interaction among students, universities and the recruiters. As result, this application provides multiple of ways implementing monetisation strategies, which include recommendations from 3rd party MOOCs and a direct channel between the students and recruiters.

The development of this application was conducted by using and following the cutting-edge technologies and methodologies. The application was created using JavaScript based frameworks on both front-end and back-end layers, which eased the learning curve and reduced overall development time. At the end project a fully featured application was produced, which exhibited smooth front-end performance, sensible UI/UX implementation and stable client-server communication.

REFERENCES

- [1] Kylie, "Building a Mobile App: Ionic Vs React Native Vs Native | Simpleweb", Simpleweb.co.uk, 2016. [Online]. Available: <https://simpleweb.co.uk/building-a-mobile-app-ionic-vs-react-vs-native/>. [Accessed: 17- May- 2017].
- [2] F. Massart, "React Native vs Ionic: A Side-by-Side Comparison | Codementor", Codementor.io, 2017. [Online]. Available: <https://www.codementor.io/fmcorz/react-native-vs-ionic-du1087rsw>. [Accessed: 18- May- 2017].
- [3] A. Aggarwal, "Ionic vs React Native – Ankush Aggarwal – Medium", Medium, 2017. [Online]. Available: <https://medium.com/@ankushaggarwal/ionic-vs-react-native-3eb62f8943f8>. [Accessed: 11- May- 2017].
- [4] S. Parunashvili, "On Scaling React Applications", Applicative 2016 on - Applicative 2016, 2016.
- [5] E. Law, V. Roto, M. Hassenzahl, A. Vermeeren and J. Kort, "Understanding, scoping and defining user experience", Proceedings of the 27th international conference on Human factors in computing systems - CHI 09, 2009.
- [6] J. Nielsen, "Enhancing the explanatory power of usability heuristics", Proceedings of the SIGCHI conference on Human factors in computing systems celebrating interdependence - CHI '94, 1994.

- [7] Google, "Introduction - Material design - Material design guidelines", Material design guidelines, 2014. [Online]. Available: <https://material.io/guidelines/material-design/introduction.html#introduction-principles>. [Accessed: 24-May- 2017].
- [8] I. Clifton, Android user interface design, 2nd ed. 2015, pp. 20-22.
- [9] R. Mehta and R. Zhu, "Blue or Red? Exploring the Effect of Color on Cognitive Task Performances", Science, vol. 323, no. 5918, pp. 1226-1229, 2009.
- [10] N. Kaya and H. Epps, "Relationship between color and emotion: A study of college students", College Student J, vol. 38, no. 3, p. 396, 2004.
- [11] Sisodiya, Pushpendra Singh. Asp.Net vs Ruby on Rails vs Django-Python vs PHP? [Online] May 2017. <https://www.linkedin.com/pulse/aspnet-vs-ruby-rails-django-python-php-pushpendra-singh-sisodiya>.
- [12] Gray, Tim. WHAT'S THE BEST RESTFUL WEB API FRAMEWORK . [Online] July 2016. <http://optimalbi.com/blog/2016/07/21/whats-the-best-restful-web-api-framework-part-2/>.
- [13] Lee, Changpil. An Evaluation Model for Application Development Frameworks for Web Applications. s.l.: The Ohio State University, 2012.
- [14] Buckler, Craig. The SQL vs NoSQL Holy War. [Online] 2015. <https://www.sitepoint.com/sql-vs-nosql-differences/>.
- [15] ELKSTEIN, DR. M. Learn REST: A Tutorial. [Online] <http://rest.elkstein.org>.
- [16] A comparison of SOAP and REST implementations of a service based interaction independence middleware framework. Mulligan, G. and Gračanin, D. s.l. : Winter Simulation Conference, 2009, December.
- [17] Francia, Steve. REST Vs SOAP, The Difference Between Soap And Rest. [Online] 2012. <http://spfl3.com/post/soap-vs-rest>.
- [18] JSON Web Token Introduction. [Online] 2017. <https://jwt.io/>.
- [19] "University of Southampton Open Data Service |Open Data Service | University of Southampton", Data.southampton.ac.uk, 2017. [Online]. Available: <http://data.southampton.ac.uk>. [Accessed: 24- May- 2017].
- [20] "Academic Courses |Open Data Service | University of Southampton", Data.southampton.ac.uk, 2017. [Online]. Available: <http://data.southampton.ac.uk/dataset/courses.html>. [Accessed: 24-May- 2017].
- [21] "Learn about Udemy culture, mission, and careers | About Us", Udemy About, 2017. [Online]. Available: <https://about.udemy.com>. [Accessed: 24- May- 2017].
- [22] Ramos, J., 2003, December. Using tf-idf to determine word relevance in document queries. In Proceedings of the first instructional conference on machine learning.
- [23] "Average Web Developer salary in UK | Web Developer salaries on CWJobs.", Cwjobs.co.uk, 2017. [Online]. Available: <https://www.cwjobs.co.uk/salary-checker/average-web-developer-salary>. [Accessed: 09- May- 2017].
- [24] Medium, T., Breakdown, T. and Becket, X. (2016). The Cost of Advertising Nationally Broken Down by Medium. [online] WebpageFX Blog. Available at: <https://www.webpagefx.com/blog/business-advice/the-cost-of-advertising-nationally-broken-down-by-medium/> [Accessed 23 May 2017].
- [25] "Higher education in numbers", Universitiesuk.ac.uk, 2017. [Online]. Available: <http://www.universitiesuk.ac.uk/facts-and-stats/Pages/higher-education-data.aspx>. [Accessed: 23- May- 2017]

APPENDIX I - BACKEND API SPECIFICATION

Endpoint	Methods	Description	Request/Response JSON body structure
/signup	POST	Create new user and get back newly created user object and auth token.	Takes: {"username": "username", "password": "password" } Returns: { "success": true, "token": "JWT ...", "user": Object }
/auth	POST	Authenticate user and get back user object and auth token.	Takes: {"username": "username", "password": "password" } Returns: { "success": true, "token": "JWT ...", "user": Object }
/user	GET	Get current user simply from the auth token. Required HTTP Header: 'Authorization: JWT ...'	Returns: { "success": true, "token": "JWT ...", "user": Object }
/modules/{:id}?fields=<field,>	GET	Get a module object either from the id. Accepts field queries, which will make the response contains only the fields specified. Required HTTP Header: 'Authorization: JWT ...'	Returns: { "success": true, "module": Object }
/modules/find/{:name}	GET	Find modules by full or partial name (search like behaviour). Required HTTP Header: 'Authorization: JWT ...'	Returns: { "success": true, "message": 'Found following modules.', "modules": [Object1, Object2..] }
/favourite/{:module_id}	POST	Favourite a module, add it to the users module list. Required HTTP Header: 'Authorization: JWT ...'	Returns: { "success": true, "message": "Module added to favourites" }
/unfavourite/{:module_id}	POST	Unfavourite a module, remove it from the users module list. Required HTTP Header: 'Authorization: JWT ...'	Returns: { "success": true, "message": "Removed added modules from the favourites" }
/feedback/{:module_id}	PUT	Leave feedback and rate a module for the module specified by the ID. Required HTTP Header: 'Authorization: JWT ...'	Takes: { "rating": "5", "feedback": "feedback" } Returns: { "success": true, "message": "Successfully added rating and feedback for the module." }
/notes/{:module_id}	PUT	Change the notes field of the module specified by the ID.	Takes: { "notes": <html code or object of the notes> } Returns: { "success": true, "message": "Successfully changed the notes of the module." }

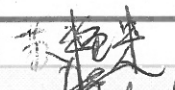
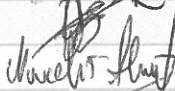
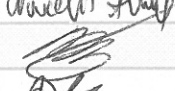
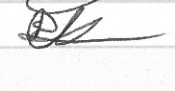
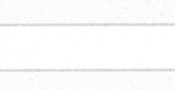
COMP6214 Coursework Marks Distribution

Return

A completed and scanned copy of this form must be submitted via ECS Handin as part of your team coursework submission

Please record your proposed distribution of the total number of marks awarded to your team. Please enter names, ECS IDs, registration numbers and the percentage of the total team effort contributed by each team member. The contribution percentages must total 100%.

Each team member must sign and date the form before submission to confirm that they agree with the proposed distribution. Only one fully-completed form per team is necessary. A scanned copy must be submitted via the ECS Handin system by the team leader, together with the conference paper.

First Name	Last Name	ECS ID	Student No.	Percentage	Signature	Date
Yanzhong	Su	ys3n15	28252543	20%		25/05/2017
Saber	Hamidi	sh30g12	25534556	20%		25/05/2017
Ahmet	Novalic	an2n16	25028251	20%		25/05/2017
Rihards	Baranovskis	rb7e15	28895401	20%		25/05/2017
Phinephas	Asare	pa4g13	26046075	20%		25/05/2017