AdBarn - Let's make Advertisements fun

Darshil Patel
Computer Science
North Carolina State University
Raleigh, North Carolina
dpatel14@ncsu.edu

Muppidi Harshavardhan Reddy
Computer Science
North Carolina State University
Raleigh, North Carolina
mreddy2@ncsu.edu

Ruthvik Mandava
Computer Science
North Carolina State University
Raleigh, North Carolina
rmandav@ncsu.edu

Srinivas Parasa
Computer Science
North Carolina State University
Raleigh, North Carolina
sparasa@ncsu.edu

ABSTRACT

People have a lot of choices for buying a product of their interest. There are a lot of websites that compares each product and guide the user to accomplish their purchasing wisely. Yet, all the major companies allocate a lot of budget towards advertising their product on television, social media, newspapers etc. People often feel very differently about the ads on social media and any streaming service as they hinder the flow of any video they are watching. Yet, companies do not stop advertising the product. There is a requirement for both the consumers and the enterprises to have a cheaper platform dedicated just for advertisement. So the service Adbard is created just for advertising where people who really needs to look at the demos of products and performance testing can navigate to Adbarn and checkout. This paper discusses about the AdBarn website created, the functionalities added, argues the need of such website and the work that has been accomplished till date and the future scope and also the outcomes of the additional features added to the existing system.

Keywords

Web Application; Advertising; paper; Software Engineering;

1. INTRODUCTION

Before buying a product, one always tries to trade off between cost and quality. People spends a lot of time in browsing the web pages that compares the products and then decide which one to buy. If they have no exposure to the features, they consult the experts or close ones to buy. But the same people do not like to be interfered with the advertisements pertaining to a single product while they were in the middle of watching a video, though they want the product. As everyone knows the effects of advertisements and the impact it has on viewers is enormous. All the multinational companies invests huge amounts into advertising, but at times without proper planning and suitable medium, it often goes waste. While online advertising has changed the world today but yet with the freedom of user ads do not reach deep within the society. Users often skip or ignore them most of the times. So instead of investing in such mediums and with this is in mind and to avoid such scenarios a new application was brought into existence to deal this specific problem. Adbarn was started to make the corporate companies stop advertising on Youtube or other streaming websites, but post it on AdBarn. AdBarn expects to impress the companies to start investing in them through increasing their user base by introducing a lot of deals to the users who are watching the advertisements and at the same time attract companies to post their ads at a very lower cost.

AdBarn is an online portal where enterprises can post their advertisements, demos, performance videos, reviews, stats, prices etc and allows user to search and browse them and at the same time maintain a user profile which helps AdBarn to show recommended ads. The users also get rewarded for investing their time in watching those advertisements and will be provided with some offers if the user decides to buy from the link provided by the entrepreneur. Rewarding for watching is always debatable. AdBarn tries to balance the finance between the rewards given to the people and the enterprises invested money which might discourage people from watching too many videos for rewards. People can view the advertisements at their convenience, read comments and views on the products and decide on what to buy. Additionally we are thinking of adding a feedback option for a user to rate or provide feedback to help advertisers assess the reach of the advertisement. It is a win win for both viewers as they get rewards, watch advertisements based on their interest and need and on the other side for the enterprises in terms of budget plans, reviews and feedback about their newly posted advertisement.

2. SYNOPSIS

2.1 Previous Work

Adbarn was implemented using the MEAN stack. Though the features implemented were very minimal, the core purpose of the website i.e streaming videos is incorporated which made us concentrate on other features that could make website complete. The website on its on home page allows to select user login or enterprise login. The user portal is dif-

ferent from the enterprise.

Users can view videos uploaded on the website and comment under the video. There is no algorithm that shows specific videos. The website just lists all the videos uploaded by all the enterprises in the order of uploaded data retrieved from the database. Users can view the tags for a video, but do not have a way to search video based on those tags.

Enterprises can upload videos to the websites after the login if they have provide coins so that they can be given to the users who watch their videos. Enterprise need to provide the tags and the description along with the title when uploading the video. They can also view some trivial statistics about their videos such as view and any comments. The website therefore successfully allows enterprises to upload videos and user to view the videos.

2.2 Software Development Cycle

For proper execution of the complete project starting with gathering required data, design, implementation and testing, an efficient SDLC has to be selected based on the project expectations. A Software Development Life Cycle is a conceptual model used in project management that describes the stages involved in an information system development project, from an initial feasibility study through maintenance of the completed application.

Therefore in our case we have decided to go with Agile Methodology since agile being an incremental software development methodology we have the liberty to keep developing based on our outputs from time to time.

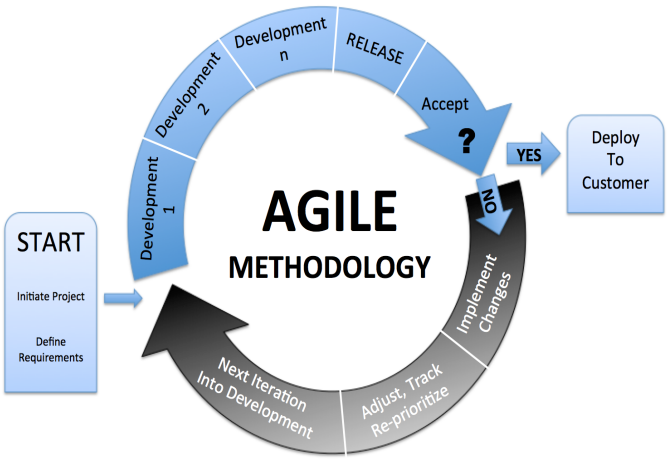

Figure 1: Development Life Cycle

Along with this we have decided to go with TLD (Test Last Development), since the scale of the application as of now is small and therefore given the time constraint we wanted to invest more time for development and if we would have gone with TFD we might end up making the development process complicated and time may have been wasted. In that case, hence to be on the safer side we went with TLD.

2.3 Initial Survey

Based on the decided project scope, we conducted a survey using Google Forms to determine the need and user's expectations for the project. The questions focused more on having recommendations, handling payments, sharing the coins, notifications and anonymity of the users from the enterprise users.

From this initial survey we have gathered the required data based on which we could focus on prioritizing the features to be developed first and then in an iterative fashion keep adding features in the given time frame and ensure a working product at the end. The majority of the users in the pre-survey considered recommendations to be a valuable addition to the system as it can be seen below.

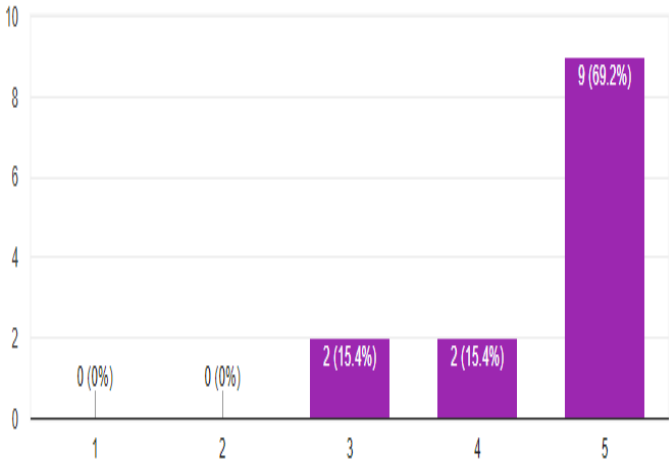

Figure 2: Requirements Gathering

Implementing the above functionality required us to add a feature where the users can like/unlike a video which they view based on which the recommendation system can provide proper recommendations to the users.

Also providing the users with a feature to add comments on to a video has to be handled carefully. Because this feature provides the users with a platform where they can express anything in their own words. At this point, it is the responsibility of the users to use the provided feature in a constructive way but being developers it is our responsibility to prevent the features of the application being used in an abusive way. That is why we allowed the users to report abusive comments.

Also, a majority of the viewers considered it to be better not to share their information with the enterprises. As per the requirements, we decided not to share the viewers information with the enterprise. The enterprise users do get to see the statistics of the videos they have uploaded but can not see the viewers information.

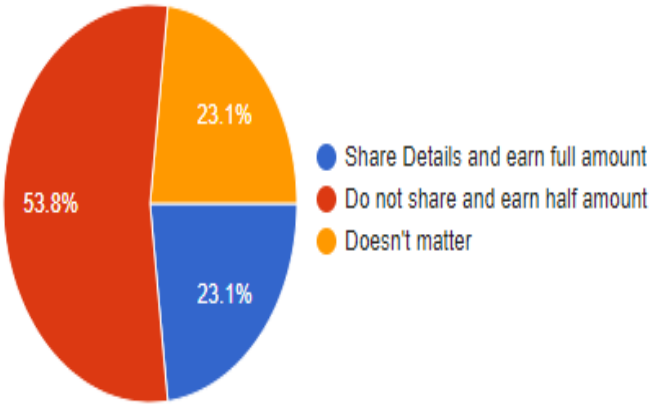

Figure 3: Requirements Gathering

2.4 Project Goals

2.4.1 *Anonymity or Profile Building*

Currently AdBarn provides viewers to login and start watching advertisements posted on the website. There is no profile maintenance or any option to select 'Wish to remain Anonymous' to the enterprise as we are trying to incorporate a feature of showing users data to the enterprise who might need it to contact them, to further promote or provide benefits.

Profile maintenance is different from being anonymous. Being anonymous is to make sure enterprises don not get any details about the viewers. Some wish not to be contacted by the representatives after they watch a products videos. They can select this option and watch the videos anonymously.

Profile maintenance on the other hand lets AdBarn develop and run Machine Learning algorithms at the back end that can help in recommending the videos or demos or performance reviews of the products that users might likes. Currently AdBard incorporated tags for every video that would be posted by the enterprise. These tags work in correlation with the likes of the users and accordingly recommend them or update them about new videos posted. We will be discussing about the notification service and recommendation service in the next sections. Therefore for the two services to work, AdBarn expects its users to provide some details about their interests and likes and at the same time make it more convenient for the users by not needing to update the search criteria every time they log in. We just want the right to track the user history, their selection of videos, the search history to start building up their profile. This would help AdBarn to start building communities that share common interests and in future can make groups to start discussing about the products either anonymously again or with enterprises directly.

2.4.2 *Notification service*

All the ecommerce websites, streaming websites do have the notification service to make sure users are notified of all the new releases, benefits, offers, coupons etc. Therefore we also felt AdBarn has to have this feature to maintain tight connection with the users. In the survey we asked users if they prefer notifications via email or a text message or just to notify them when they log in to AdBarn. We received a balanced view on the notifications and therefore we will go ahead and add the option in the user profile where the users can select their preference.

Notifications can be of different type based on the profile the users are building. They may reflect the recent history of the users or any preselected enterprises that the user selected or any previous demos that user watched that they likely want to be notified of next version released. Also it can notify users about the benefits they can select with the current coins they earned. Currently there is no feature that talks to user in any way. So notification service would be the first one to start establishing a contact between AdBarn and the users.

2.4.3 *Recommendation Service*

Streaming websites has a lot of traffic and is always crowded with millions of posts, messages, comments, reviews, notifications etc. But a single user consumes only a very tiny amount of that. How can one achieve to show users what they might like next to watch, among the huge number of posts and updates on the website. A good recommendation service is needed for such a feature to be added. Amazon reported that 38% of its sales are from recommendations. Netflix reported that $\frac{2}{3}$ of its rented movies are from recommendations. Therefore AdBarn would be incomplete without having a recommendation service. This recommendation service would suggest viewers to watch ads which are related to the watch history of that particular user.

2.4.4 *Different services for enterprises*

Currently AdBarn provides a login page for enterprises, upload option to post an advertisement, add coins which adds coins into the account of the enterprise which can be used to offer to the viewers who watches the ads. We would like to divide the services into two tiers at different cost. Tier 1 offers enterprises to view the user details who watched their posts and videos from which they can benefit by contacting them to offer discounts or explain them or schedule a demo to accomplish selling their product. Service 1 also offers more demographics data in the form of graphs or raw data that enterprises can use in their business meetings to change their strategies to concentrate more on the community where they have less reach or concentrate on a specific group where they have more confidence to sell their product. Tier 2 wouldn't be having all these features and therefore will be the cheapest option to the enterprises. Tier 2 expects the enterprises to just add coins for attracting the users to watch their advertisements.

2.4.5 *Better graphical representations for enterprises*

Currently AdBarn shows one Pie chart that showcases the number of views to a single video posted by the enterprise. Based on the service chosen by the enterprise, we would like to improve the UI to display more graphs that can help enterprise to target the viewers to contact easily. Providing raw data would attract less enterprises to subscribe to the service. Therefore our goal is to add more UI options to view different type of graphs that can help in targeting specific group of people.

2.4.6 *Sharing Coins with other viewers*

Currently the viewers can earn coins by watching the videos and those coins remain with them. There might be

cases when a particular viewer might want to share a part of his coins to other viewer who is a friend. There might also be cases when a particular viewer decides to upload advertisements and use his earned coins as a source to give the coins to other viewers who watches his advertisement. For such cases, sharing of the coins becomes a very important feature for the application. Moreover, as seen in the literature survey, none of the other similar applications provide this feature and hence it can help AdBarn to get an upper hand as compared to other applications present in the market.

3. DESIGN ARCHITECTURE

For the implementation of this web application we decided to continue implementing the backend using Node js. With the backend support from node js the frontend was developed using Angular js. Here express provided a middleware where the frontend could talk to the backend using REST APIs. For the database support of this application we have decided to go with a NoSQL database, mongodb. The combined design implementation of these technologies is straight forward as the following diagram below. The major advantage of using this MEAN stack is the consistency of the programming language and also the help it provides in integrating the different components into one. We have also used D3.js as a charting infrastructure which provides visual statistics to the enterprise. The recommendation system is completely developed in Python + Spark and it acts as an independent system from the application.

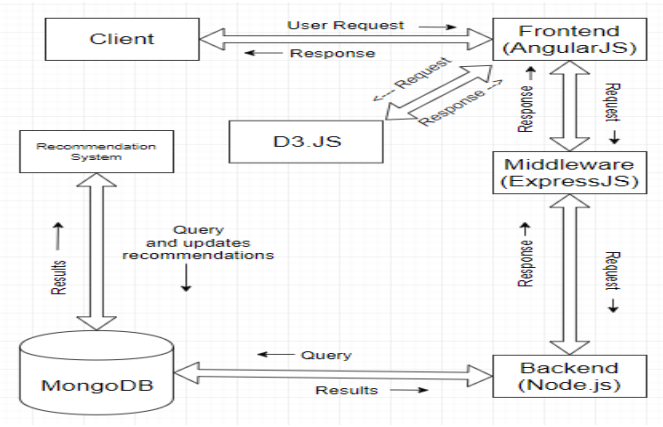

Figure 4: Design of Application

Each advertisement does not have an equal demand to be streamed. Therefore having replicas in the database for each video would be a costly inefficient design. Thanks to content delivery network or content distribution network (CDN) which is a geographically distributed network of proxy servers and their data centers. The goal is to distribute service spatially relative to end-users to provide high availability and high performance. There are two modes of transmission of a video[1] to the end users where the user can either download or stream the video. AdBarn will be a streaming portal. Users are expected to stream always instead of download as it helps us to give rewards and at the same time make sure users stay on AdBarn all the time instead of downloading.

4. TECHNOLOGIES

4.1 MongoDB

MongoDB is a free and open-source cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with schemas. MongoDB is considered to be the best NoSQL database that can be easily integrated with Node.js. There are three collections that are so far created and maintained to run the website. One stores all the user credentials, along with the videos viewed, videos liked and also the videos to be recommended to them which are generated asynchronously by the recommendation engine shown in the architecture. One for enterprises that stores their credentials, their coins and the videos that they upload. Other collections include encoded videos information that are stored as files which have mappings to the streaming platforms that provide the video content to the users. Fields in a MongoDB document can be indexed with primary and secondary indices which helped efficient retrieval of the videos that are liked by the user to show it to the enterprises and also efficient retrieval of the videos based on tag search. Also MongoDB provides high availability which is very important for any streaming application through replication and also high fault tolerance with load balancing taken care of.

4.2 AngularJS

For the front end development we have Angularjs which helps to build the applications very fast. It reduces the development time as it allows the developers to use directives which are basically reusable HTML code. If there is a set of HTML tags which have to be used at multiple places there is no need to use the same set of tags at multiple places, instead we just create a directive containing those tags and use the directive everywhere. This is similar to functional programming of the HTML DOM. Also, Angularjs allows real time DOM manipulation which was required at multiple places in the project. Moreover, Angularjs also allows us to use module bundlers like webpack which builds and bundles the whole Angulajs code into one single source file. We do not need to link newly created javascript files into the index file because webpack takes care of that.

4.3 D3.JS

D3.js is a library which is used to produce data visualizations in the form of graphs. This allows the enterprises to check which of their uploaded videos were viewed how many number of times at a given point of time. In contrast to other libraries, D3.js allows great control over the final visual result which is why this library is preferred over other libraries. Moreover, input to D3.js can be given in the form of JSON which makes it easy to use over other libraries.

4.4 Node and Express

A viewer through the client interface sends in various kinds of requests be it login or getting a video for playing, liking or un-liking the video, commenting on a video, getting the recommendations, claiming the coins and many others. Similarly the enterprise users also uses a lot of requests to the backend. All these requests are passed to the Nodejs backend using the expressjs middleware in the form

of a Representational State Transfer (REST) API. Nodejs has an inbuilt support to npm which is one of the biggest package libraries javascript has. Also, with the support to npm and expressjs, node serves as a good backbone for the system.

4.5 Python + Spark

The algorithms chosen to incorporate recommendation service will be detailed in the next sections. Here we will discuss about the technologies that we used. After many discussions we chose Python+Spark to recommend which are completely out of the MEAN stack we were discussing about. Initially we used Raccoon, an easy-to-use collaborative filtering based recommendation engine and NPM module built on top of Node.js and Redis. The engine uses the Jaccard coefficient to determine the similarity between users and k-nearest-neighbors to create recommendations. This module is useful for anyone with users, a store of products/-movies/items, and the desire to give their users the ability to like/unlike and receive recommendations based on similar users. This option was very easy for us to integrate as we had a goal to add like/unlike any video already. But collaborative filtering implemented by the machine learning community of spark is more adaptable to millions of users and videos. Using a big data framework is a must to form a complex algorithm and model over millions of users and videos. Therefore Python and Spark are incorporated into the architecture that acts as an asynchronous service that provides recommendations to the users.

5. FUNCTIONALITIES

Since our application is web based we gave instructions for the user to install the application in github ReadMe. The application can be run locally by the user. These are very simple commands that can be run by the user in limited time. Before using the application, the user has to register on the website which would help us to maintain sessions. After registration the users can login with their credentials.

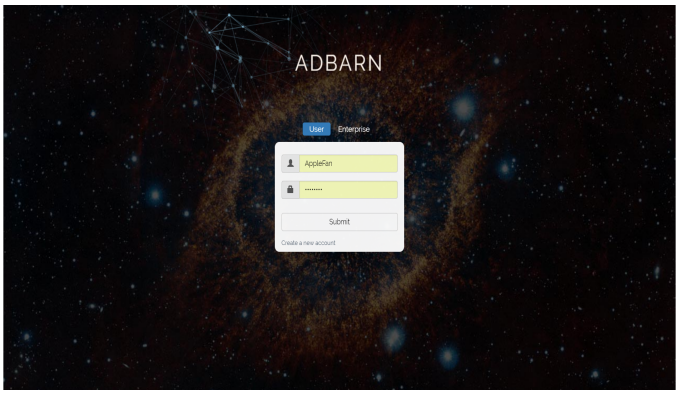

Figure 5: Viewer/Enterprise Login Page

During registration we had implemented multiple checks where the database schema and validations in place wouldn't allow two users having the same username.

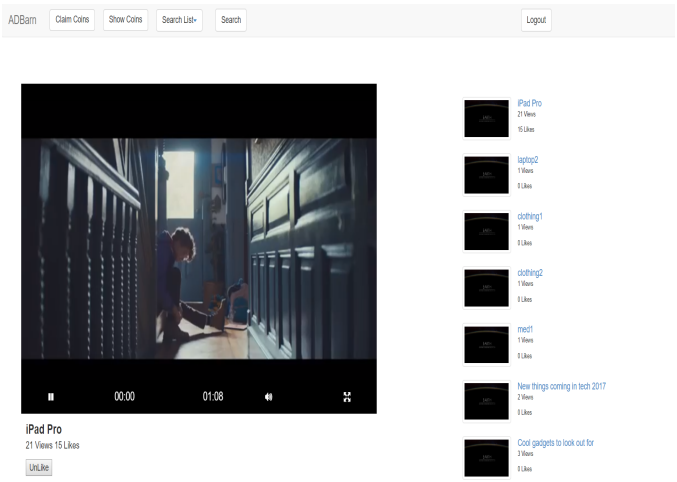

Figure 6: Viewer Home Screen 1

After logging in successfully, the viewer will come to above screen where they have an option to select the video they want to view. Also they have a drop down which can be used for the tag based searching. They also have an option to Unlike or Like a video. The following image shows that they also can add comments to the video and can also see a list of recommended videos. This list of recommended videos are retrieved using the python + spark service.

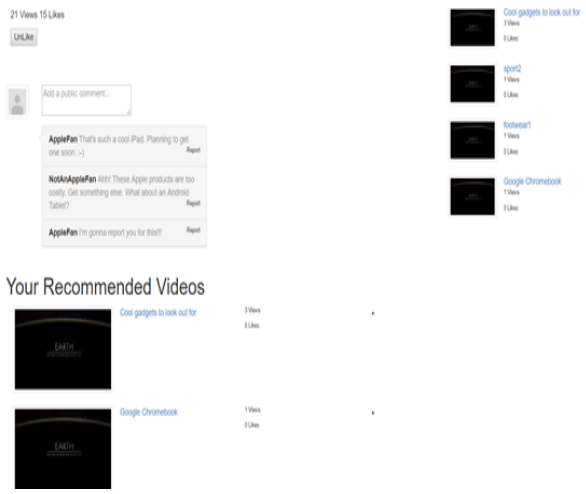

Figure 7: Viewer Home Screen 2

As far as the enterprise users are concerned, they get to see the below screen as their home page. Here they have options to Add Coins, Update the number of coins a user gets if he watches their advertisements, they can also upload new videos from here and can also view the statistics. They also have an option to delete a video which they have uploaded previously.

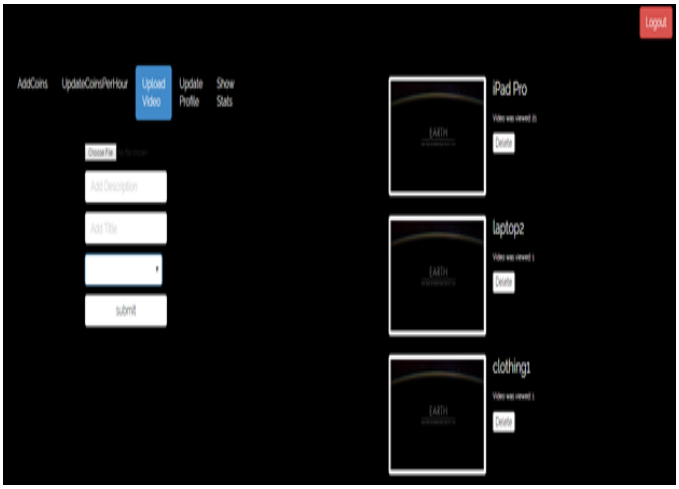

Figure 8: Enterprise Home Screen

The enterprise users can see the complete statistics of the views of their videos. For example, if an enterprise user has uploaded 10 video with each having some number of views they can see a pie chart having created with fragments of each video with the number of views deciding the size of the fragment.

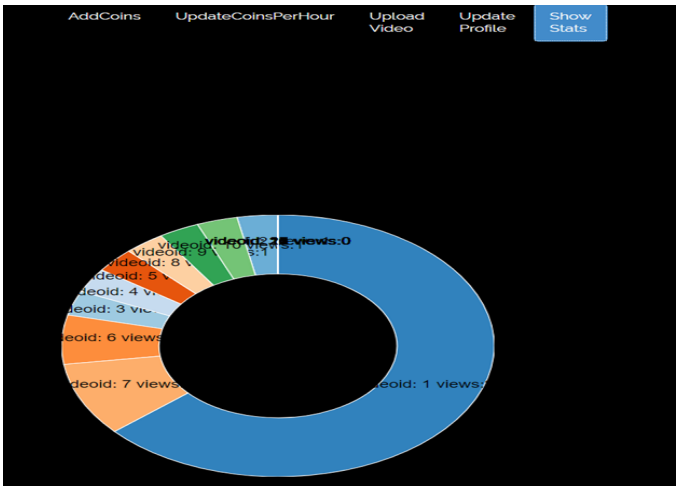

Figure 9: Enterprise Statistics

This charting has been done using D3.js. D3 has an extensive support for visualization and we can easily extend the current service to embed different types of charts. We can add conversions from one chart to another for the same purpose which can make data visualization very easy for the users.

6. IMPLEMENTATION

6.1 Like and Dislike a video

As described earlier, the like and dislike functionality was added to provide the strong foundation for the recommendation service if we would have used Racoon or likely.js for getting the recommendations. But due to the reason described in the upcoming sections, we decided not to use any of these and use a completely different recommendation engine on our own. Like and Dislike still is a helpful functionality for the users to understand what are the videos which they might like and could be fun watching them if they want to earn more coins.

6.2 Report Abuse

Report abuse was added just to prevent any abusive comments added to a video. When the users are provided with full flexibility to express their views in the form of comments, it is their responsibility to use it well but in case the users are not using proper language, it comes to the application level where they should be able to remove any comments which are reported as abusive comments.

6.3 Recommendation System

We believe the data is very sparse to start recommending videos to the users because relative amount of advertisements posted on AdBarn is very high compared to the videos watched by a single user. But collaborative filtering with ALS tuning of the parameters can help recommending even if the user watches 10-20 videos in the pool of millions of advertisements. It simply works on the assumption that you may like this video because there are a group of users that share same history or preferences with you. But simply matching the history or preferences does not scale the system as it requires to find K nearest neighbours in the pool of millions of users all the time and finding the similarity measures.

Therefore Collaborative filtering helps to scale this issue by finding out latent hidden factors between the users and the items that actually made them watch them the video or like the video or rate the video etc. Simple click on the thumbnail can be of use to start recommending. Hence this method makes automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating). The underlying assumption of the collaborative filtering approach is that if a person A has the same opinion as a person B on an issue, A is more likely to have B's opinion on a different issue than that of a randomly chosen person. For example, a collaborative filtering recommendation system for Adbarn could make predictions about which advertisement a user should like given a partial list of that user's tastes (likes or dislikes). Note that these predictions are specific to the user, but use information gleaned from many users. This differs from the simpler approach of giving an average(non-specific) score for each item of interest, for example based on its number of votes.

There are two packages that can be easily integrated into AdBarn as they are npm modules namey likely.js and raccoon.js. Likely.js is reliable on forming a matrix with n * m as dimensions where n = number of users and m = number of videos and each element in the matrix represents rating or number of times the user has viewed. This package could

not scale on a distributed service and also terribly fails when there are few users and millions of videos. Raccoon on the other side could solve this problem through storing information about likes and dislikes of the user but uses a complete different db system REDIS. It is redundant to store all the user information here as well and also has to solve the issue of consistency.

Therefore we chose Apache Spark to deal with this issues. Spark MLlib is a distributed machine learning framework on top of Spark Core that, due in large part to the distributed memory-based Spark architecture. But it will be a difficult problem to synchronously call the recommendation service whenever the user logins as that requires to allocate some K nodes to start building the model and predict the ratings and then recommend. Therefore we made it an asynchronous stand alone service that runs in every specific period of time selected and fill in the recommendations to the user in his profile. The code captures the information directly from the MongoDB, forms the best accurate model through cross validation technique and uses recommendAll(int n) method available to recommend to all the users, 'n' number of recommendations which are stored into the user collection in the MongoDB with no other collection created. These recommendations are shown on the website to the user below the video they are currently watching.

6.4 Pause video on lost focus

Pausing the video when the tab or the window gets inactive was one of the important features we wanted to add to the application. Without this feature, the viewers can just mute the video, go to some other tab or window and do any other work and still earn money without actually watching the advertisement. This completely breaks the purpose for which the application was created. To prevent this issue, we decided to add a functionality where the video gets automatically paused as soon as a viewer switches to some other window or tab. The application also resumes the video once the application tab is back to active. This ensures the enterprise users that the advertisements are actually viewed by the viewers to earn the coins.

7. SURVEY RESULTS

Now with modifications and embedding more features we wanted the users view of this application to understand the reach-ability and applicability of this application in real world. So we started of with scenario simulations and asked users opinions in such a case for much better and reliable evaluation results.

7.1 Enterprise Point of View

Firstly we wanted to test how efficient and easy our application was to use with the new features added and therefore we asked the users to follow a set of operations while using the application. But before that we have two point of views in this application. Therefore we asked the user to initially look it in the point of view of an enterprise and execute the set of operations as shown below.

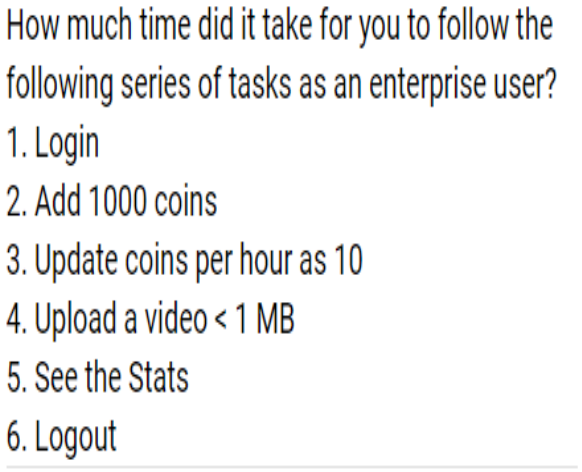

Figure 10: Enterprise Simulation

From the user's response we observed that most of users stated that these set of operations took around 1.5 - 2 Mins. From this we could understand multiple factors about the application like how easy it is to adapt to the interface. Whether it was easy for the user to understand the operations available to the enterprise. Hence with such effective evaluation format we could deduce how efficient the application is based on the time taken by the user. Therefore a timeframe of around 2 mins tells us that most of the users were able to easily understand and adapt to the application.

7.2 Viewer Point of View

Secondly as we said the application needs to be seen from two point of views we now asked the users to perform a set of operations which are related to viewers and at the same time let us know the time taken by them to finish those set of operations. We covered all the major features and implementations which are necessary for any viewer to be aware of while using this application. So combining such operations gives us the complete picture of the adaptability of this application in all ways.

After performing these operations users stated that it took them around 2.5 mins to complete the operations and very few of them stated it took them around 3 mins. The users for whom it took around 3 mins probably had difficulty in understanding and adapting to the functionalities in the application. While most of them could complete them in around 2.5 mins which tells us that most of them found it to be easily understandable, since we performed the same operations and it took us around 2 mins, but since we were the developers it was easy for us to understand the first time and given that, the fact that users could complete it in 2.5 mins proves that the application was indeed easy to understand.

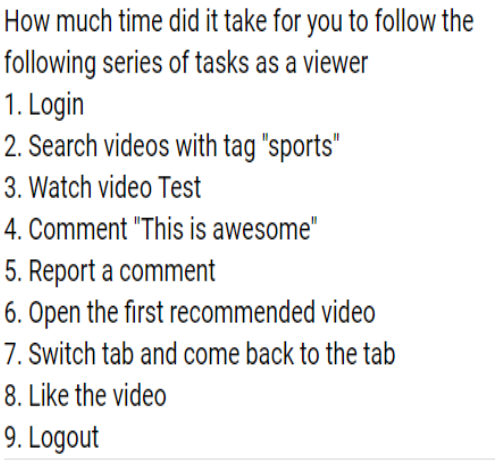

Figure 11: Viewer Simulation

7.3 Efficiency of Recommendation System

One of our important added feature is the the recommendation system, so we wanted to know how relevant the users felt the recommendation was and generally get an idea how much accuracy does this model hold with the users. So for that we have asked the users to rate the recommendations provided for which we have received impressive results with almost 75% users saying rating it between 4-5. But for this we simulated a scenario for the users to evaluate with having various topics and asked them to watch the videos under technology and asked them specifically if they could come across any other technology related videos with the recommendation and based on this users rated as we can see above.

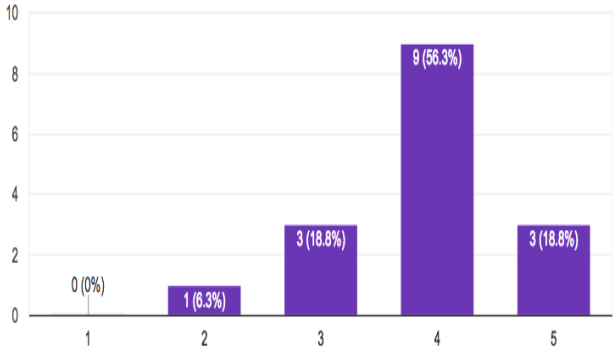

Figure 12: Efficiency of Recommendation System

7.4 Usability of Like/Unlike

We also wanted to test one other feature implemented which was like and unlike buttons, this gives an idea to the enterprises also to evaluate themselves as a company. To ensure efficiency and reliability of this service we asked the users to test the like and unlike options and we had almost 94% of users stating that they had no issues at all with respect to like and dislike options which ensures us that the enterprise evaluation results would be accurate and will give a broader image.

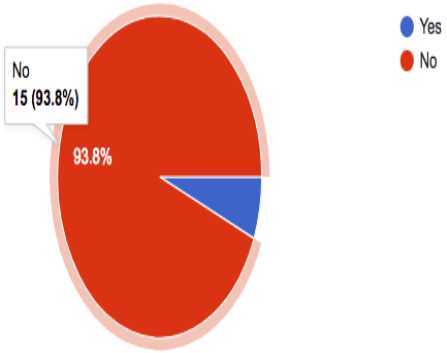

Figure 13: Usability of Like/Unlike Features

7.5 Working of Auto Pause

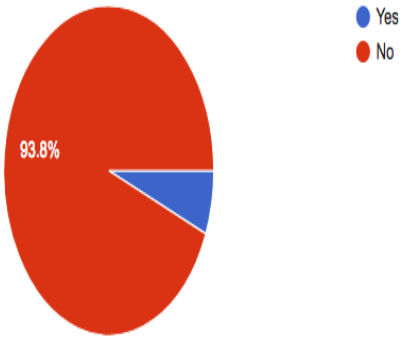

Figure 15: Auto Pausing of the video

One other important aspect of such an application is credibility so as we said we added a feature which doesn't allow the user to go off from viewing the ad and if someone does

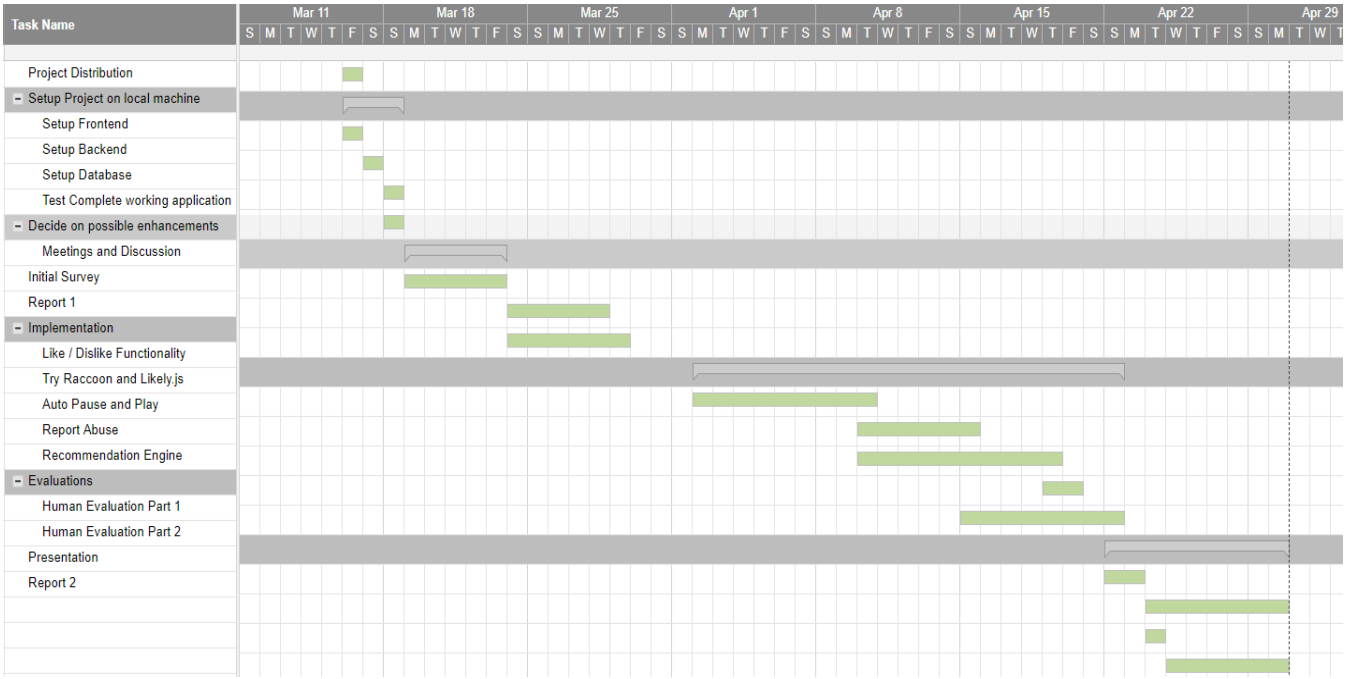

Figure 14: Project Timeline

so, the video automatically pauses. So we asked the users to specifically simulate a scenario where they would be watching a different video on some other site or working on any application. Along with this the users were then asked to logon to adbarn and start watching video and while the video was running the users were asked to shift from viewing it to some other application or browsing some other page. By doing so they had to check whether the video playing on Adbarn is paused. After simulating the scenarios almost 94% of the users found out that the video was automatically being paused which actually prevents from users exploiting the application.

With all these implementations end of the day the feasibility of such an application in the business point of view is very essential and therefore users after using the application had the basic idea of what the application has to offer. So we asked the users, further what kind of implementations would ensure applicability of this application in the real business world and we obtained a few quality suggestions. Few of users pointed out that limiting the ads viewed per user per day would definitely prevent from users exploiting the application. At the same time we had few users suggesting that the capping the coins a user can earn in a week or month depending on the available ads at that point of time, which is indeed true and in the business point of view it actually ensures a healthy applicability of such an explanation. Finally a few users expressed concern regarding the complexity the application includes with respect to an enterprise and pitching such an application and making an impact needs a lot proof of profitability which becomes one of the most challenging issue for the application to be implemented.

8. PROJECT TIMELINE

On receiving the project, we decided to follow a specific timeline and updated our deadlines as required. Following is a Gantt chart of the timeline which was followed for the project.

As it can be seen, we had a part of the evaluations done before the presentation and conducted a second evaluation cycle for the project once the presentation was done.

9. ACKNOWLEDGEMENT

We would like to express our sincere gratitude towards our Professor Tim Menzies and our mentor Ken Tu who have guided us throughout our project and helped us with their valuable suggestions. We would also like to thank our fellow peers who have given us their valuable insights from time to time which enabled us to develop this application in an efficient way.

10. CONCLUSION

While adding new functionality to the application, we tried to go through similar applications and see what are the funcitonalities those applications are providing. We tried to implement the positives from all these different applications and avoid the negatives they had. This was done so that AdBarn can tackle the competition well if released to a larger number of users. It also allowed us to gauge the level of functionalities we support as compared to similar applications. Below is a table showing the comparison chart of similar applications like Swagbucks[2], InboxDollars[3], Perk.com[4], QuickRewards[5] and CreationsReward[6] with AdBarn.

	Disabled Fast Forward	Full Screen View	Video Load Time	Auto Pause & Play	Reco-mmend-ations	Like / Unlike Video
AdBarn	Yes	Yes	Fast	Yes	Yes	Yes
Swagbucks	No	Yes	Fast	Yes	No	No
InboxDollars	No	No	Slow	No	No	No
Perk.com	No	Yes	Very Slow	No	No	No
QuickRewards	No	Yes	Fast	Yes	Yes	No
CreationsReward	No	Yes	Fast	Yes	No	No

Figure 16: Comparing Similar Apps

11. REFERENCES

[1] Shyam Katta, Santhosh Dharmagadi, Sai Harsha Suryadevara, Sai Sri Harsha Kanamanapalli *(2018). AdBarn: Unique Way of Earning Money through Advertisements*

[2] *http://www.swagbucks.com/watch*

[3] *https://www.inboxdollars.com/videos*

[4] *https://perk.com/bonus/video*

[5] *https://www.quickrewards.net/*

[6] *https://www.creationsrewards.net/earn/watch-videos*

[7] S. Shunmuga Krishnan, Ramesh K. Sitaraman *Understanding the Effectiveness of Video Ads: A Measurement Study*

[8] Yuxue Jin, Sheethal Shobowale, Jim Koehler, Harry Case *The Incremental Reach and Cost Efficiency of Online Video Ads over TV Ads*

[9] *https://www.emarketer.com/Article/Click-Rates-Complicate-Online-Video-Ad-Metrics/1008493*

[10] Dapeng Wu, Yiwei Thomas Hou, Wenwu Zhu, Ya-Qin Zhang, Jon M. Peha *Streaming Video over the Internet: Approaches and Directions*

[11] *Streaming Databases for Audio and Video Theory and Praxis*

12. CHITS

1. XWV
2. ZQZ
3. YQS
4. IZG
5. LGC
6. HXR
7. THQ
8. BWY
9. KWV
10. RMH
11. CFX
12. FTQ
13. SFL
14. XVE
15. ZKZ
16. JPX
17. DNR
18. BWG