Blowfish

By

Harry Kennedy, Calvin Mays,
& Cody Spurlock

Submitted to
The Faculty of the School of Information Technology
In Partial Fulfillment of the Requirements for
The Degree of Bachelor of Science
In Information Technology

Harry Kennedy	4/18/16 Date
Calvin Mays	4/18/16 Date
Cody Spurlock	4-18-16 Date
James Scott - Faculty Advisor	4-18-16 Date

The University of Cincinnati
College of
Education, Criminal Justice, and Human Services
April 2016

Table of Contents

List of Figures

Abstract
Blowfish is an app that combined the most popular features of Snapchat and Yik Yak and created a new social media app soon to be available on Android and iOS platforms. The app functions anonymously, so the user is not required to maintain a user account. Anonymous interaction is based upon the user's geographical location. If the user is within 1.5 miles of our predefined location point, the user will see all of the content posted within that radius. If the user is greater than 1.5 miles away our predefined location point, then the user will not see posted content. The features of Blowfish are focused on taking and posting pictures and videos to a timeline. Users within the 1.5 mile radius can see this timeline. Users also have the ability to comment on all content that is posted. Additionally, we have included a Like/Dislike rating system so popular content and comments are approved by the user base, and we have included a top post timeline, so the most popular posted are viewed by the greatest likes.

Problem

Snapchat is a photo/video based sharing app where users have the ability to take photos, record short videos, and edit them. Users can share this content with friends where it disappears after up to ten seconds. Also, users can post this content to a “story” where it can be viewed by all of a user’s friends for twenty-four hours. But with snapchat, it isn’t possible to post content to a public story where content is viewed by people in the area. It is also not possible to comment or rate content posted to a story the user’s friends puts up. There are accounts out there that are maintained to serve the purpose of a public story, but it is also against Snapchat’s terms of use to do so. Yik Yak is an app that is anonymous and a timeline of posts that can be viewed are based on your location. The issue with Yik Yak though is that it is based on short messages and it allows photos, but there are heavy restrictions on what type of photos can be posted. These two social media apps are just one of many that has these issues. There have been other anonymous apps out there that have been shut down due to Cyber bullying, because they were not managed properly. In currently existing social media applications, there aren’t features available that allow a user to anonymously submit, post, and view photos/videos with other users in a specific geographical area without violating service terms.

Solution

The solution to this issue was simple; we developed a whole new social media application. The application combined the features of existing social media apps. The different combined features gave the application a completely different user experience. More specifically, we have combined the popular features of Snapchat and Yik Yak to create a new social media application. The features of the app include a custom camera for taking photos and recording videos, an option to post the content to a public timeline, view the public timeline, the ability to comment on content posted, rate content and comments with some type of like/dislike system. The timeline a user can post to and view is all determined by a user's geographical location. The app does not require a user to login, but the app does use a unique identifier on each device to manage users. This will keep everything anonymous among users. The new application is deployed on the two most popular mobile platforms, iOS and Android using Adobe PhoneGap, Apache Cordova, and the ionic framework. The backend of this app is an infrastructure built around node.js, express, and PostgreSQL.

User Profile

Figure 1 shows our user profile in regards to who is likely to use the application.

User Profile Form
Application: ***Blowfish, a social media app***
Potential Users: ***College students***
Software and Interface Experience: ***Mobile application interfaces on iOS and Android platforms***
Experience with Similar Applications: ***Yik Yak, Snapchat, and Twitter***
Task Experience: ***The ability to take pictures and short videos, submit pictures, view, comment, like/dislike pictures and videos***
Frequency of Use: ***Multiple times daily***

Figure 1. User Profile

Gantt chart

Figure 2 shows our Gantt chart which is the schedule that will be used throughout project completion.

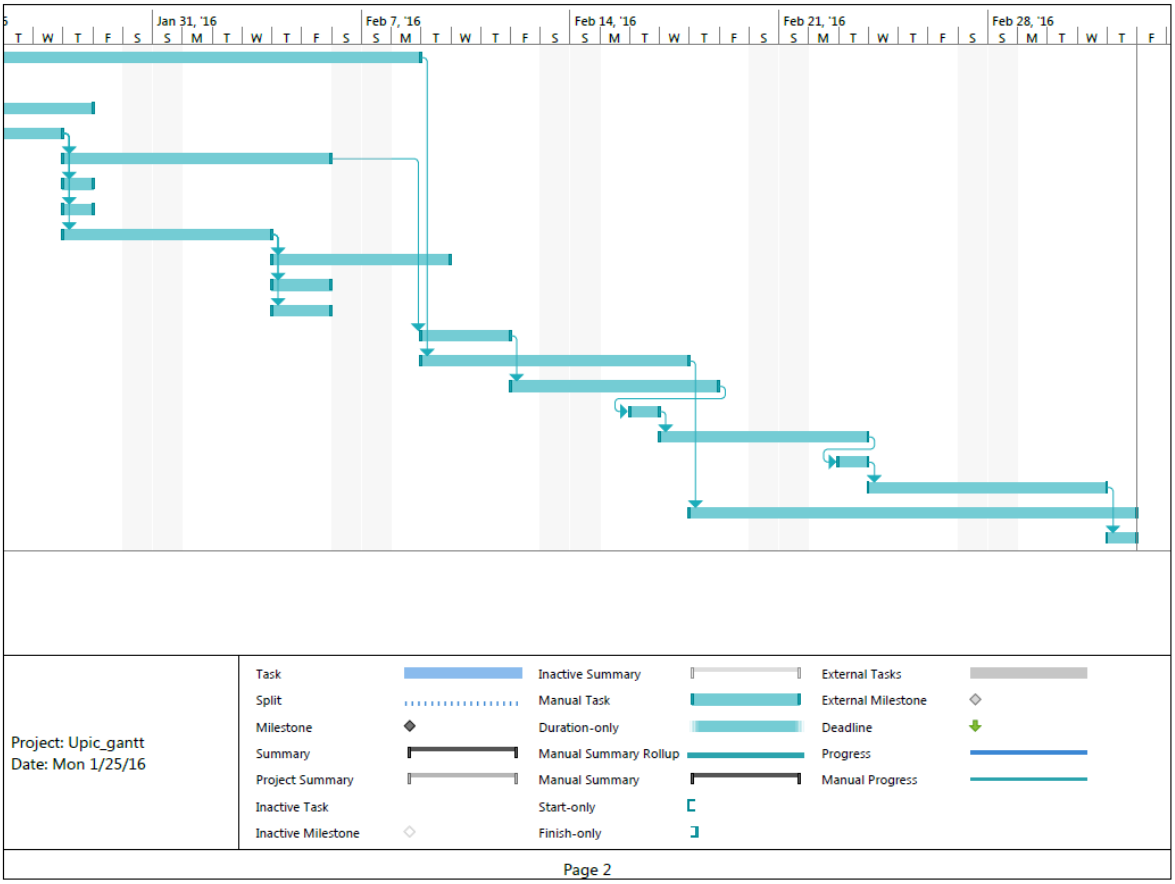

Figure 2. Gantt chart

Use Case Diagram

Figure 3 shows the use case regarding a standard user and an administrative user.

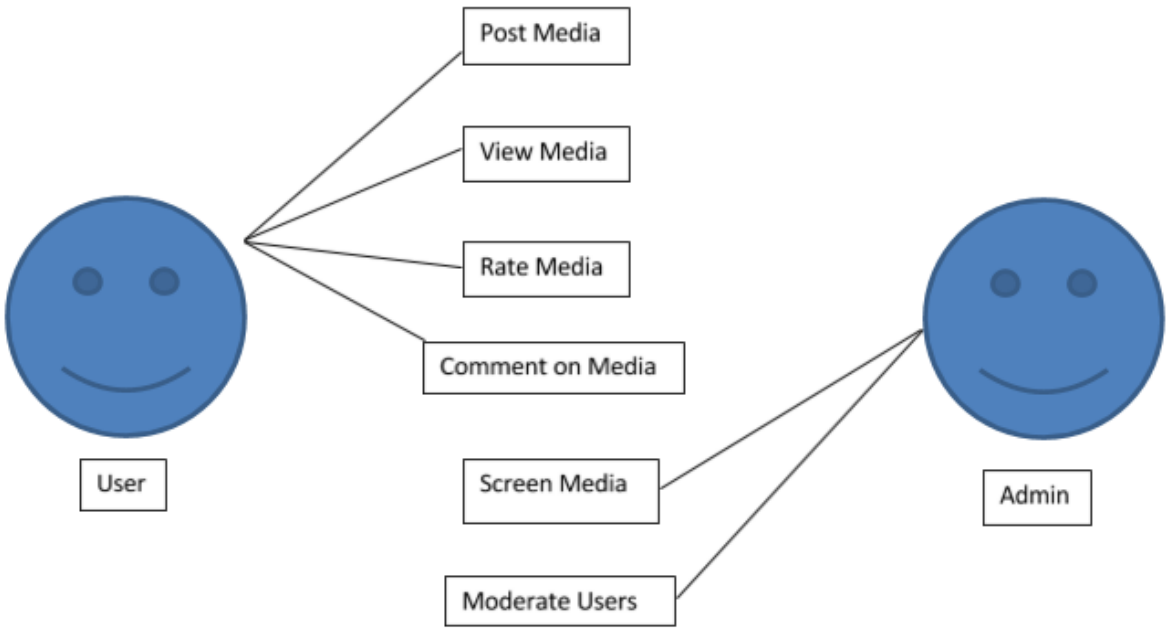

Figure 3. Use Case Diagram

Budget

Throughout the entire project, our group did not incur any cost. The hardware and infrastructure utilized was provided by UC's CECH Sandbox. For the software, we used open source software that did not require licensing fees. In the spring semester, we budgeted one-hundred dollars. This was planned to be used for the migration of our servers from the CECH Sandbox to a Cloud service provider such as Amazon Web Services (AWS).

Testing Client-Server Communication

We have planned and detailed the results of our client-server communication tests that have taken place, and we have documented the results of tests. We tested the functionality of our app (Blowfish), the stability, and the user-interface of the app.

System Requirements for Blowfish

Below are the basic system requirements for the functional portion of the app, the stability portion of the app, and the user-interface portion of the app.

Functional Requirements:

1. The app should allow users to post pictures and videos
2. The app should allow users to post comments on the pictures and videos
3. Users should be able to rate content (Like/Dislike)
4. Users will only be able to post and view all content if they are within the set location boundaries

Stability Requirements:

1. The app must run on Android platforms
2. The app must run on iOS platforms

User-Interface Requirements:

1. The app should have at least 3 different windows:
    a. The custom camera/video recorder
    b. The Timeline window of all posts
    c. The Timeline window of top posts
2. The app must also be user friendly

Test planning

Our team held weekly meetings where we discussed our progress and where we were as a group in regards to meeting our deliverables. As it stands, we mostly followed the project schedule below, but have delayed the next deliverable or began the next deliverable early based on progress discussed in our meetings. Tests were performed weekly by the Infrastructure Architect and Software Developer. The Systems Engineer set up systems as needed.

Deliverables

Task Name	Duration	Start	Finish
Phase III: Implement	13 days	Thu 1/21/16	Mon 2/8/16
Set up "test_db"	1 day	Fri 1/22/16	Fri 1/22/16
Code local functions	5 days	Fri 1/22/16	Thu 1/28/16
Add Queries to test_db	3 days	Mon 1/25/16	Wed 1/27/16
Test client/server communication	7 days	Thu 1/28/16	Fri 2/5/16
Add Comment Table to test_db	1 day	Thu 1/28/16	Thu 1/28/16
Add location table to test_db	1 day	Thu 1/28/16	Thu 1/28/16
Implement Google Maps API server-side	5 days	Thu 1/28/16	Wed 2/3/16
Implement Identity Management	4 days	Thu 2/4/16	Tue 2/9/16
Implement SSL security	2 days	Thu 2/4/16	Fri 2/5/16
Implement post timelines	2 days	Thu 2/4/16	Fri 2/5/16
Implement Prod. DB	3 days	Tue 2/9/16	Thu 2/11/16
Phase IV: Testing	7 days	Tue 2/9/16	Wed 2/17/16
Fine tuning/tweaking	5 days	Fri 2/12/16	Thu 2/18/16
First Beta release	1 day	Tue 2/16/16	Tue 2/16/16
Distribute app for bug testing	5 days	Wed 2/17/16	Tue 2/23/16
Upic 1.0 release	1 day	Tue 2/23/16	Tue 2/23/16
Tweaks/fixes	6 days	Wed 2/24/16	Wed 3/2/16
Phase V: Release	11 days	Thu 2/18/16	Thu 3/3/16
Upic 1.0 Final release	1 day	Thu 3/3/16	Thu 3/3/16

Figure 4. Deliverables

In order for our group to meet the deliverables, we planned to test functionality, stability, and the user-interface. Each of the tests below must pass to actually meet our deliverables. We as a group have passed 8 out of the 10 Functionality test requirements and both of our Stability and User-Interface testing requirements.

Functionality Test Requirements:

1. Remote DB connectivity
2. Test DB Data can be queried returning results
3. Test data returned in JSON format
4. Test data can be stored in DB via HTTP request

5. Test data can be queried and returned in JSON via HTTP Request
6. Functional data can be stored in DB via client request
7. Functional data can be retrieved via client request
8. Location data can be stored
9. Data can be stored/retrieved based on location

Stability Test Requirements:

1. App runs on Android platform
2. App Runs on iOS platform

User-Interface Test Requirements:

1. App is user-friendly
2. App is visually appealing to users

Test Data

All of our test data is shown in the figures below:

Blowfish Testing Report (Functionality)							
Req. (No.)	Test Case	Input	Expected Output	Actual Output	Pass/Fail	Reason	Test Date
1	Remote DB connectivity	H.Kennedy	Successful Connection	Connection Denied	Fail	Local Firewall policy, PG config file not set	1/27/2016
		H.Kennedy		Connection Allowed	Pass	Disabled Firewall, Set config to allow remote connections	1/27/2016
2	Test DB Data can be queried returning results	H.Kennedy, C.Spurlock	Results of query	Errors	Fail	Incorrect Syntax	1/27/2016
		H.Kennedy, C.Spurlock		Results of query	Pass	Correct Syntax	1/28/2016
3	Test data returned in JSON format	H.Kennedy	Results of query in JSON	Results of query in JSON	Pass	Data was printed in JSON format	1/28/2016
4	Test data can be stored in DB via HTTP request	H.Kennedy	View test data in the database	No test data in the DB	Fail	Incorrect Syntax	2/4/2016
		H.Kennedy		Test data in the DB	Pass	Correct Syntax	2/5/2016
5	Test data can be queried and returned in JSON via HTTP Request	H.Kennedy	Test data returneed in JSON format	No data. success code: 200 OK	Fail	When declaring a variable, the defined query was placed in the method	2/8/2016
		H.Kennedy		Test data returned in JSON	Pass	Removed method when defining query	2/8/2016
6	Functional data can be stored in DB via client request	H. Kennedy, C. Mays	JSON response of 'Data stored!'	no error	Fail	HTTP post requests methods were not using multipart/form data	2/9/2016
				JSON response of 'Data stored!'	Pass	HTTP post requests methods were using multipart data	3/16/2016
7	Functional data can be retrieved via client request	H. Kennedy, C. Mays, C. Spurlock	Data returned in JSON	Values returned as "[object], [object]"	Fail	Data was not defined on the client side	3/16/2016
				Data was returned as JSON	Pass	Data was defined on the client side	3/22/2016
8	Location data can be stored	All	N/A	N/A	Fail	Did not implement methods for storing location data	n/a
9	Data can be stored/retrieved based on location	H. Kennedy, C. Mays, C. Spurlock	Data tagged with location id returned	Data tagged with location id returned	Pass	Complex functions were implemented into the code base	4/4/2016
10	Data is transmitted/received securely	All	N/A	N/A	Fail	Did not implement secure data transmission	n/a

Figure 5. Functionality Testing Report

As the figure above shows, we as group decided not to store received location data to meet requirement 8. The only location data we stored was the coordinate points of the University of Cincinnati. We also did not implement transmitting encrypted data for time and scope purposes, so requirement 10 was also not met.

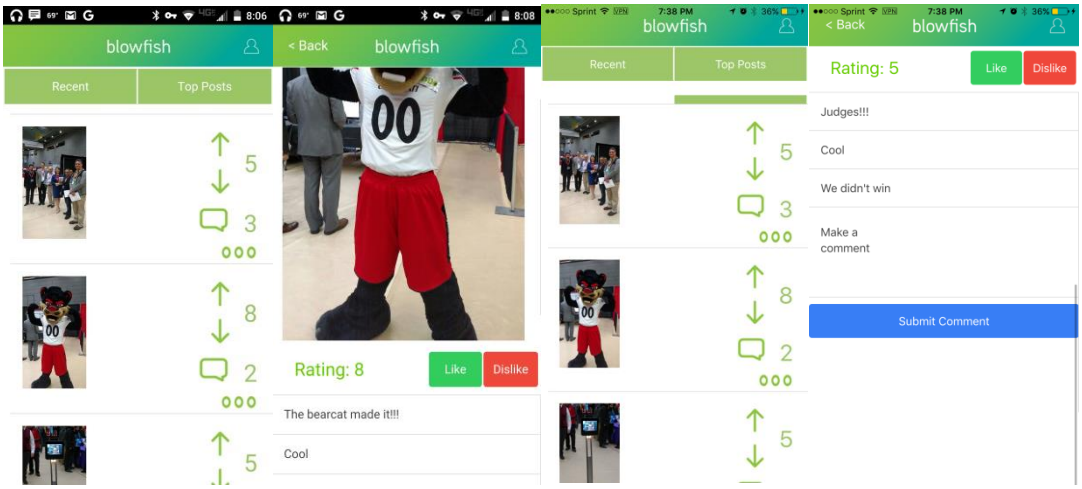

Figure 6. User-Interface/Stability testing visuals

The figure above shows the visuals of our user-interface and stability test requirements. The visuals show the app running on both Android and iOS platforms. We have passed all of our Stability testing requirements. The visuals also show a simple interface that is user friendly and also visually appealing. We have passed all of our User-Interface testing requirements.

Conclusion

By providing a social media app on the two most popular mobile platforms, we have provided our users with a way to view and share different experiences with others. We have constructed a new server-side infrastructure using node.js and PostgreSQL as we transitioned away from BaaSbox in late December 2015. Our client-side development used PhoneGap Cordova, and the Ionic framework. We have made our deployment and maintenance quick and easy with the potential for higher scalability in the future. We are aiming to improve overall security in our infrastructure. We are also geared towards a tested and stable release of Blowfish 1.0 to the App and Google Play stores sometime in the near future.

References

Alvord, Jamie. "New App Exposes Inappropriate Material to Students." *The Herald*. N.p., 23 Feb. 2015. Web. 2 Nov. 2015.

Betters, Elyse. "What's the Point of Snapchat, and How Does It Work?"Pocket-lint. N.p., 15 Sept. 2015. Web. 2 Nov. 2015.

Larson, Selena. "What Secret's Shut down Means for Anonymous Apps."*The Daily Dot*. N.p., 30 Apr. 2015. Web. 2 Nov. 2015.

Pullen, John Patrick. "You Asked: What Is Yik Yak?" Time. N.p., 4 Feb. 2015. Web. 2 Nov. 2015.

Sevilleja, Chris. "Build a RESTful API Using Node and Express 4." Scotch. N.p., 15 Apr. 2014. Web. 2 Nov. 2015.