Purview Design Document

Please direct all communication to: ml25@princeton.edu

Gordon Chu	gachu@princeton.edu
Yannis Karakozis	ick@princeton.edu
Matthew Li	ml25@princeton.edu

¶ Section 1. Overview

Purview aspires to become the anonymous social platform for short-form video content-sharing for university students and high schoolers. It allows users to create video content associated with the location of the user and specific tags the user selects. All videos are up to 10 seconds long and get automatically removed from public view after 3 days.

The videos produced are viewable by all users in the nearby area, who can upvote or downvote each video. Videos can be searched by tags and they are presented in descending order of net score (upvotes - downvotes). Purview also provides an automatic filtering functionality that filters out videos that receive above a threshold level of downvotes to ensure the content presented is not offensive or inappropriate.

Users have access to all their current active videos, as well as their top three most voted posts of all time. The user is also presented with the Hall of Fame of their area, which includes the five most upvoted active videos produced in the user's region.

¶ Section 2. Requirements and Target Audiences

Our generation's obsession with social media is unparalleled -- there seems to be an app for just about anything you could think of, each of which has firmly cemented themselves within social lives of people around the world. Want to share a status? Use Facebook. Want to share a picture with your friends? There's Instagram. Want to share a picture with your friends but only for 10 seconds? Snapchat. These applications are all identical in essence: users create and share content, but simply meet this goal with different approaches. These examples just go to show that even small nuances in functionality can still draw immense interest.

There is also another feature of some social media applications that has become quite popular among university students and high schoolers: anonymity. Most notably, the location-based social network known as Yik Yak quickly gained popularity within university campuses around the nation simply by appending this concept to an otherwise plain post-sharing application. Combining anonymity with automatic content filtering in the form of upvotes and downvotes, Yik Yak eventually became a popular source of student-generated content -- until it began enforcing usage of identity. The fall of Yik Yak presents to us the opportunity of creating a new application that takes advantages what Yik Yak had done right.

So where does Purview come in? What makes Purview different from currently popular apps is its unique mixture of the various successful elements of other applications. Given that anonymous social applications are extremely popular among university students and high schoolers and given the obsession of those demographics with Snapchat, Purview is just the app these demographics would use. It combines the anonymity of Yik Yak and the visual focus of the Snapchat Stories features, both of which are extremely popular among the target user base. It appeals to the obsession of millennials with personal story sharing, and enables them to get informed about what is going on, while also guaranteeing their privacy.

As none of the key elements of our application are novel by any means, it is not unimaginable that apps with a similar mix of these features have already been attempted. Indeed, there is an app called Panama developed in 2015 that has almost the exact same features. However, the fact that it has not become viral after one year and a half from deployment allows us to not consider it an 'successful solution'. Therefore, we have a good chance of introducing this application as a fresh platform for private anonymous video-sharing. To some, the failure of Panama may cast doubts on the growth potential of Purview, but with our focus on university students and the advantage of being based at Princeton, a strategy that starts by developing school-wide prevalence could allow us to generate sufficient initial traffic to continue expanding.

¶ Section 3. Functionality

This section delineates the two typical use case scenarios of Purview: 1) a user making a video post, 2) a user viewing video post feeds. Both use cases start as follows:

A user loads Purview and logs in automatically to their account (or signs up by using their email and by picking a password for their account if they open the application for for the first time). The user then is transferred to the landing screen, which is the video feed of their area. At the top of the screen, there is a search bar. At the bottom of the screen, the user is presented with a Navigation tab of three buttons: “Post”, “My Feed”, “My Profile”. The “My Feed” button is highlighted, indicating the user is at the screen of the feed of his area.

3.1 Use Case 1: User Posting a Video - Cindy

Cindy taps the “Post” Button and is transferred to the camera interface. By clicking at the button at the bottom of the screen, she starts recording. After seven seconds, she taps the button again and stop recording. The replay of her video comes up. She reviews the replay of the video and decides she does not like the result. Thus, she cancels the recording and re-records. This time she does not tap the stop button before the ten seconds have elapsed, so the camera stops recording automatically after 10 seconds.

Cindy is satisfied with her latest recording. She types a tag at the text field that has popped up at the middle of the screen. While she is typing the tag, popular tags in the area appear below the text field in the form of a dropdown menu. Cindy fancies the tag “TI” and selects that one. She completes the post by clicking the “Post” button. A message that her post has been made successfully pops up, along with the buttons “View ‘TI’ Videos”, “Post Again” and “My Profile”.

Cindy clicks on the “My Profile” button. Her account screen comes up. There, she sees the video she just posted at the top of the “My Active Videos” list and also at the second cell of the “My Top 3” list. Each entry is a table field comprised of the video thumbnail, the time the video was posted and the video’s net score. Her video has already gotten net 10 upvotes. She clicks on the thumbnail of her most recent recording and views it again. After the video stops playing, she is returned to the “My Profile” screen. Satisfied with her most recent creation, Cindy closes Purview and goes back to partying responsibly at TI.

3.2 Use Case 1: User Video Post Feeds - Bob

Bob types at the search bar field the tag videos of which he would like to view. While he is typing, popular tags in the nearby area appear below the text field in the form of a dropdown menu. Bob is curious what is going on at “TI” this Saturday night and selects that one.

Bob is presented with the feed of the “TI” tag. The videos are presented in descending order of their score, with the ones of the best score being presented at the top. Each entry is a table field comprised of the video thumbnail, the time the video was posted and the video’s net score.

Bob selects the top video. The video expands and is played in full-screen mode. Bob enjoys the full 10 seconds of a guy chugging a bottle of orange juice at TI. After the video is completed, Bob is automatically returned to the “TI” feed screen. He upvotes the video and the score of the video is updated to +1 of its current score in real time.

Bob then clicks the “Back” arrow at the top of the screen and returns to the main feed. He scrolls down and clicks on the fourth video on the list. Bob quickly realizes this video is a recording of the same exact scene. Because he does not want to waste time re-watching this video as he has to finish Assignment 5 of COS333 which is due in 45 minutes, Bob taps on his screen while the video is playing. The video stops and Bob is returned to the main feed screen.

Feeling the urgent need to write more test cases for the COS 333 assignment, Bob closes Purview and goes back to coding.

3.3 Non-Goals

This version will *not* support the following features:

- Changing and redeeming passwords
- Playing music from any music streaming application, while recording.

¶ Section 4. Design

4.1 Three-tiered Design

The design will follow a three-tiered architecture consisting of a front-end mobile application, back-end web server, and persistent database in order to implement the desired functionality described in the previous section.

4.2 Client/Server Interface

The client-facing interface provided by the web server will be a RESTful API exposing the Videos and Users services. API keys will be used for logging and authentication to the API.

4.2.a Videos API

The Videos API will allow users to search, play, upload, and edit videos limited to their geolocation. These features will be accomplished via the HTTP verbs GET, POST, PUT, PATCH, and DELETE and JSONified data. Below are a couple sample requests and their results.

Search through all videos ordered by upvotes

```
GET /videos HTTP/1.1
Host: api.projectname.com
```

```
HTTP/1.1 200 OK
Content-type: application/json

{
      “videos”: [video1, video2, ...]
}
```

Query Strings

- *lat* - latitude
- *long* - longitude
- *tag* - the tag you wish to filter on, may occur multiple times in query string if multiple tags filtered on
- *limit* - number of results to be returned
- *offset* - how many results to skip

* note: the video objects returned by this query will not have the video inside them, but a thumbnail instead, in order to minimize processing time and response sizes. If rendering the video is desired, a separate GET request to the Videos API is necessary (see below).

Request a video with a given id

```
GET /videos/{video-id} HTTP/1.1
Host: api.projectname.com
```

```
HTTP/1.1 200 OK
Content-type: application/json

{
        “video_id”: 5,
        “video”: [bytes],
        “tags”: [“cos”, “friend”],
        “user_id”: 63,
        “timestamp”: 1489509323,
        “location”: format TBD,
        “upvotes”: 4,
        “downvotes”: 2,
}
```

Upload a video

```
POST /videos HTTP/1.1
Host: api.projectname.com

{
        “user_id”: 27,
        “location”: format TBD,
        “tags”: [“charter”, “friday”],
        “video”: [bytes]
}
```

```
HTTP/1.1 200 OK
Content-type: application/json

{
        “video_id”: 9001
}
```

4.2.b Users API

Find information on a specific user*, **

```
GET /users/{user-id} HTTP/1.1
Host: api.projectname.com
```

```
HTTP/1.1 200 OK
Content-type: application/json

{
        “user_id”: 7777,
        “username”: “deadbeef”,
        “score”: 88,
        “achievements”: [achievement1, ..],
        “videos”: [video1, video2, ...]
}
```

* note: if the authentication token shows that the user making this request is not the user whose user id is in the url string, a limited subset of this information is returned - only user_id, username, score, and achievements
** note: the video objects returned do not store the entire videos, only thumbnails (see the Videos API for an explanation)

Create a new user*

```
POST /users HTTP/1.1
Host: api.projectname.com

{
        “password”: “bro”,
        “email”: “joe@princeton.edu”
}
```

```
HTTP/1.1 200 OK
Content-type: application/json

{
        “user_id”: 11111,
        “username”: “soelite”
}
```

* note: validation will be performed both client and server-side.

4.3 Server/Database Interface

The database will persistently store all the data regarding users and videos. Videos will be deleted 3 days after being uploaded. The database will respond to queries in the SQL format from the server only. The database will not be publically available.

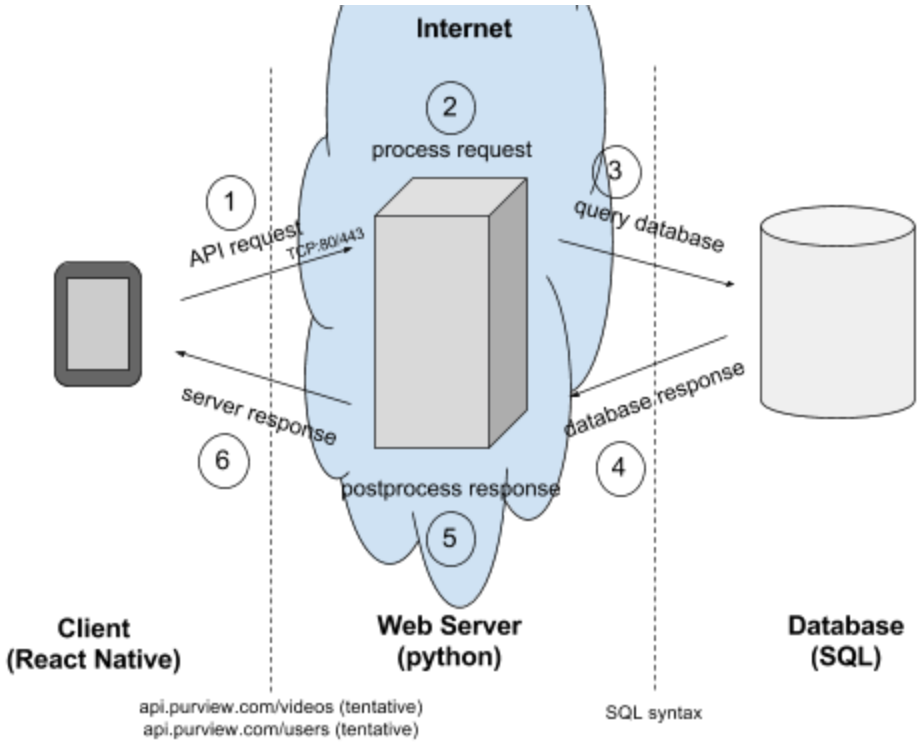
4.4 A likely implementation

Now that a reference architecture and its interfaces are established, the following selection of technologies has been chosen to implement them:

Tier/Responsibility	Technology	Framework
Mobile application	JavaScript	React Native 2.0.1
Web server	Python	Flask, SQLAlchemy
Database	MySQL/Postgres	--
Protocol	TCP	--

The mobile application will be coded using React Native for portability. Research shows that among similar applications (e.g. SnapChat, YikYak, Instagram), Python is a commonality for the web server. Since the back-end fetches data from the database and returns it without much processing, the lightweight Flask library will suffice (as opposed to the hammer that is Django). For the database, the application data can be modeled well relationally, and therefore SQL suits this need aptly. Finally, even though our application will be sending videos around quite often, TCP was chosen over UDP as a protocol because (i) each individual video will be fairly short (<10 seconds) and (ii) there is evidence that video over TCP can work if pre-fetching is done properly (e.g. YouTube).

Below is the final implementation architecture with specific technologies filled in.



¶ Section 5. Timeline

Due Date	Feature/Milestone	Front-end Work	Back-end Work
3/24/17	End-to-end connectivity with API (dummy data)		● Set up Flask app ● Set up Flask-autodoc ● Set up Flask-testing
3/24/17	Deploy site	● Code up project status page	● Acquire domain name and hosting
3/24/17	User creation and user persistence (including auth)	● Login flow ● Sign up flow ● Landing page stub ● Update status page	● Define User schema ● Set up FlaskAuth on all API calls ● Store and return actual data for Users API calls
3/31/17	Video upload	● Integrate camera ● Update status page	● Define Video schema ● Implement POST Videos API call (store data)
3/31/17	Video search	● Search box ● Search results display ● Update status page	● Implement GET Videos API call
4/7/17	Voting system	● Thumbs up/thumbs down on each video ● Update status page	● Implement PUT/PATCH Videos API calls ● Implement safe-search/results
4/7/17	Hall of fame	● Hall of fame page hooked up to API calls ● User can see top 3 videos and all active videos ● Update status page	● Implement hall of fame API calls ● Implement top 3 videos per user
4/14/17	Slippage time	● Documentation ● Testing	● Documentation ● Testing
4/14/17	Prototype	● Prepare for demo	● Prepare for demo

		● Update status page	
4/28/17	Slippage time	● Incorporate feedback from prototype session ● Documentation ● Intuition testing with users	● Incorporate feedback from prototype session ● Documentation ● Intuition testing with users
4/28/17	Alpha phase	● Update status page	
5/5/17	Beta phase	● Update status page	
5/14/17	Project complete	● Update status page	

¶ Section 6. Risks and Outcomes

6.1 Risks from Division of Labor

Due to our team's distribution of experience between the different components of the project (i.e. some members have mostly front-end experience, while others mostly back-end experience), the division of labor will also generally adhere to the three-tier architectural divisions. As such, potential risks and delays may arise from the communication between developers of different components.

For example, delays could arise from inconsistencies in the type and format of requests and responses sent and received between components; however, such delays should not occur given proper planning. More plausible setbacks could originate from issues related to each team member's lack of experience with unfamiliar components of the project, such as what can be feasibly implemented within other components, as well as the relative difficulty and time needed to be accomplished. Such misunderstandings can cause difficulties in assigning project-wide feature deadlines and milestones.

6.2 Potential Technology Delays

Although our team has collective experience in developing in JavaScript with React, as well as in mobile app development, React Native represents new territory for us. As such, we do not yet have a comprehensive grasp of what can be easily implemented with this technology. Therefore, some development time should be allocated to probing

the specifics of this technology. Despite these potential unknowns, we are confident in this technology for the front-end due to its extensive documentation and the well-known applications that have already been built on it. If for some reason this approach is not viable, then we can always switch to the native approach using Swift and Xcode for development. This backup plan would sacrifice the portability of our project, but assures our capability to implement the application.

Specific to iOS development, only one member of the team owns a Mac, and only two of the three own iPhones. This would cause certain limitations in development, simulation, and testing of the front-end component. However, since the front-end specialist is the owner of the Mac, it does not represent a big drawback. Furthermore, there are Mac computers available on campus that would allow us to develop the mobile component on multiple machines if necessary.