Cinemus Development Members:

* Preston Billion Polak – pbillionpola2019@fau.edu
* Madison Verger - mverger@fau.edu
* Jorge Risco - riscoj2018@fau.edu
* Clifford Ariel – cariel2016@fau.edu

|  |  |
| --- | --- |
|  | Milestone 3 Project Proposal and High-level Description |
|  |  |
|  | Project Group 12 (“Cinemus Development”)  CEN4010 Principles of Software Engineering, Summer 2020  7/14/20 |

History:

Ver. 1 -–- 6/16/20  
Ver. 2 --- 6/30/20  
Ver. 3 --- 7/14/20

# Executive Summary

Cinemus (short for Cinema Us) will be an online platform for people to watch YouTube videos with other people together in real time; it is aimed to cater to those who used to go to movie theaters with friends, but now cannot due to quarantines.

Besides the primary purpose of filling the role of real-life movie theaters, Cinemus will also serve as a platform to meet people with similar interests, as well as highlight and share lesser-known content creators on YouTube.

It will be structured similarly to the streaming platform Twitch, and users watching videos together will be able to communicate within the site using an open text chat. Users will be able to create and join various “rooms,” and can either join “popular” rooms with many people in them, search specifically for their friends’ rooms, or join rooms recommended to them based on their previous rooms.

# Competitive Analysis

|  |  |  |
| --- | --- | --- |
| Competitor | Similarities to Cinemus | Key differences to Cinemus |
| Twitch (streaming site) | Both are platforms where multiple viewers can watch content and talk with each other live and in real time.  Both have multiple “rooms” that viewers can host and join. | Twitch’s content comes from a single person (the streamer), while Cinemus’ comes from an outside source (YouTube). This lessens any delays between a host of a room and their viewers. |
| YouTube livestreams | Both are platforms where multiple viewers can watch content and talk with each other live and in real time.  Both host YouTube content. | YouTube livestreams are meant for creators to get new content to viewers instantaneously, while Cinemus allows users to watch existing YouTube videos together. |
| Netflix Party | Both are platforms where viewers can watch preexisting content together, and both aim to simulate the experience of a movie theater. | Netflix Party allows users to watch movies available on Netflix, while Cinemus hosts YouTube videos. |

# Data Definition

* Cinemus users may create and join **theater rooms** (or simply **rooms**) to watch content. Each room may play one YouTube video at a time.
* The video player in each room is **synchronized** to all users such that when any viewer changes the video or segment of video being played, that change is reflected for all viewers. Any difference between users’ players is known as **desynchronization**, or simply **desync**.
* The creator of the room is known as the **host** and may change the initial settings of the room upon creation.
* All other users in a room are known as **viewers** and may have varying amounts of control of the room (depending on the settings that the host set).
* Each room may have one or more **directors** who may have more **permissions** than other viewers, such as editing the video queue, or using the text chat. The host will always be a director, but the role of the director may pass between other users, depending on the settings of the room. The directors will also always have strictly more permissions than other viewers.
* Each room will have a **queue** of videos set to play. When one video finishes, the next video in the queue will begin to play. The queue may be accessed and managed by directors and viewers, depending on the settings of the room.
* Each room may have a **text chat** (or simply **chat**) that users may type into to create public messages to the room. The chat may be accessible to all users, or may not, depending on the settings of the room.
* Among the room’s settings, the host may assign some number of **tags** to a room upon creation. These may describe the genre of video planned to play, the size of the room, or other attributes. When searching for public rooms, viewers may filter their search to only include rooms with specific tags.
* Each room shall have a **thumbnail** visible to users browsing through rooms on Cinemus’ homepage. This thumbnail will be the currently playing video’s default YouTube thumbnail, updated approximately every ten seconds (in case the played video changes).
* All users will have some number of other users that they are **friends** with. The list of these other users is known as the **friend list**.

# Overviews, Scenarios, and Use Cases

## Scenario 1:

Alice, Bob, and Charlie are friends who live apart and are stuck at home. Before the quarantine, they had plans to go to the nearest theater to watch a film but now cannot do so due to lockdowns. They agree to instead use Cinemus from home as a substitute.

Once they create Cinemus accounts, Charlie creates a theater room and waits for Alice and Bob to join. While creating the room, he ticks a box to make the room “private” so that only users on Charlie’s friend list can see the room. Once the room is created, Charlie sends a link (outside of Cinemus) for the new room to Alice and Bob. Alice clicks the link and a new tab opens on her browser into Charlie’s room. Meanwhile, Bob is already on the Cinemus homepage, and so he navigates to the “Friends’ Rooms” tab to find and enter Charlie’s room. When both have entered, Charlie begins playing the video that he had already chosen and added to the queue prior.

When creating the room, Charlie also changed other settings of the room so that all viewers would be able to add videos to the room’s queue. Once in the room, Alice and Bob use the text chat to add their favorite short films into the video queue, and then begin to talk with each other and Charlie in the chat.

## Scenario 2:

Dave is stuck at home between semesters with nothing to do, so he decides to open up Cinemus. Not looking for anything in particular, he browses through the Popular Rooms tab on the homepage and sees multiple rooms with many people on them.

Dave can see the tags and thumbnail of each room, but nothing he sees particularly appeals to him, so he switches to the Search tab and selects two tags to filter results: Comedy and Short Videos. Dave also filters the rooms found to only show those with closed queues.

With a more specific selection of rooms, Dave finds and joins a public room that seems appealing and begins talking to the many viewers in the text chat. One other viewer in particular gets along with Dave and shares his interests, so Dave sends a friend request on Cinemus and they exchange contact information in the room’s text chat.

# High-Level Functional Requirements

1. Cinemus shall be able to host multiple separate rooms, each with one video player synced to all viewers. Users shall be able to create their own rooms at any given time.
2. When creating a room, users shall be able to modify the settings of the room to allow for various changes and customizations.
   1. Users shall be able to mark rooms as either “public” or “private.” Public rooms shall be visible to all users, while private rooms shall only be visible to users on the host’s friend list.
   2. Users should be able to choose whether time-based tags are enforced in videos in their room (see requirement 2f).
   3. Users shall be able to modify how the role of the director is handled. Options should include:
      1. “Host only”: only the host has the role of director.
      2. “Assigned”: the host can give and revoke the role of director to any user in the room.
      3. “Random”: the role of director shall pass between viewers (selected randomly) at set time intervals.
   4. Users shall be able to modify what permissions the viewers and directors each have. Any permissions granted to viewers shall also be applied to directors. Permissions should include:
      1. Adding to the queue
      2. Removing videos from the queue
      3. Pausing and playing the current video
      4. Reading messages in the text chat
      5. Sending messages in the text chat
   5. Users shall be able to enable and disable the text chat.
   6. Users shall be able to choose up to five premade tags to attach to the room (see requirement 4c).
      1. Most tags shall describe the genre of video being played in the room (such as “Educational” or “Animation”), but there shall also be tags describing the length of the video (such as “5-15 minutes”). When creating a room, the host may choose at most one of these length-based tags, so as to prevent contradictions.
      2. Genre tags shall not be enforced in a room. For instance, a “Non-Animation” tag will not prevent animated videos from being added. However, time tags should be enforced, depending on the room’s settings (see requirement 2b).
3. Within a room, users with the proper permissions shall be able to add and remove videos in the room’s queue. The queue shall be visible to all users, though only some may have permissions to edit it.  
   1. Users with the proper permissions shall be able to add videos by pressing a button in the room, creating a pop-up. From here, there should be two methods of adding a video:  
      1. Entering the YouTube video ID of the video they want to add.
      2. Entering text in a search box, and choosing a video from the results of the search. This search shall use YouTube’s Search API.
   2. Users with the proper permissions shall be able to remove videos from the queue by pressing a button next to the video.
4. The homepage shall have several “tabs” that allow the user to find rooms in various ways. These tabs should include:
   1. A “popular” tab that displays all public rooms, sorted by descending amount of viewers present.
   2. A “friends” tab that displays all rooms created by people in the user’s friend list. This shall show both public and private rooms.
   3. A “recommended” tab that displays a selection of rooms based on the tags in rooms that the user has recently entered.

There shall also be a “search” bar that allows the user to enter text. The search bar shall display all public rooms whose title matches the search term(s) used. The user shall be able to select up to five tags to filter their search by.

While browsing through rooms in any of these areas, the user shall be able to see the thumbnail, tags, and host name of each room.

1. Each user who signs up shall have an account with various items attached to it:
   1. A unique username that will appear next to messages they send in text chats and next to rooms that they host
   2. A password of at least 8 characters that contains at least 1 non-alphabetical character, not visible to any user (these passwords shall be hashed and salted before being stored)
   3. An email account, not visible to any user
   4. A friend list, not visible to any users besides the account holder.
2. Users shall be able to send friend requests to other users. These requests should be initiated from within the site, such as by searching for the recipient’s username in the user search (see requirement 8).
3. Users shall be sent alerts through the site when they have received a friend request from another user, as well as when a user on their friend list creates a new room. These alerts shall **not** be sent to the user’s email.
4. The site shall have a “user search” bar that allows users to search for the username of any other user, and from there send a friend request to that user (see requirement 6).
   1. Given an entered string of text, this search shall simply return all users whose usernames contain the string.

# List of Non-Functional Requirements

1. Security: All passwords shall be hashed and salted before being stored on the LAMP server.
2. Speed: For any two users with similar connection quality, the video player’s desync *shall* not exceed three seconds and *should* not exceed one second.
3. Portability: The site *shall* work on the latest versions of Google Chrome and Mozilla Firefox, and *should* work on the latest versions of Microsoft Edge, Internet Explorer, and Safari.
4. Ease of Use: Users completely new to Cinemus (but experienced with computers) should express satisfaction with the site’s UI and design.
5. Reliability: The daily downtime of the site shall not exceed 5 minutes more than that day’s downtime of FAU’s LAMP server.

# High-Level System Architecture

## System Architecture:

The system is structured in a layered manner, where each layer communicates with its adjacent layers, and does not often communicate with others.

The layers in this system, from “frontmost” to “endmost,” are the user interface, the HTML and JavaScript code, the PHP code, and the database. See UML Diagrams for a depiction of the layers and their communications.

## Database Organization:

The MySQL database used shall have five tables. Note that all attributes are stored as up-to-20-long series of characters, unless noted otherwise.

1. The User table stores account data for Cinemus users, and has four attributes:
   1. A unique, plaintext username
   2. A hashed and salted password, stored as a 20-long series of characters
   3. A plaintext email, stored as a up-to-255-long series of characters
   4. The ID of the room the user is currently in (from the Room table)
2. The Room table stores data for all rooms, and has six attributes:
   1. A unique room ID
   2. The name of the room
   3. The settings of the room, stored as a number that encodes the binary/ternary options
   4. The current number of users in the room, stored as a number
   5. The username of the host of the room (from the User table)
3. The Tag table stores the list of different available tags, and has only one attribute: the name of the tag.
4. The Usertag table stores the tags of the rooms that the user has entered, and has two attributes:
   1. A username of a user (from the User table)
   2. A name of a tag (from the Tag table)
5. The Roomtag table stores the tags associated with each room, and has two attributes:
   1. An ID of a room (from the Room table)
   2. A name of a tag (from the Tag table)

## Media storage:

Most stored data will be stored in the MySQL database, as detailed above. However, the queue of videos associated with each room will be stored in a “flat file” on the server. These files will be in the form of a comma-separated list of YouTube video IDs. Each user’s friend list and friend requests will also be stored in flat files.

## Tools:

Development:

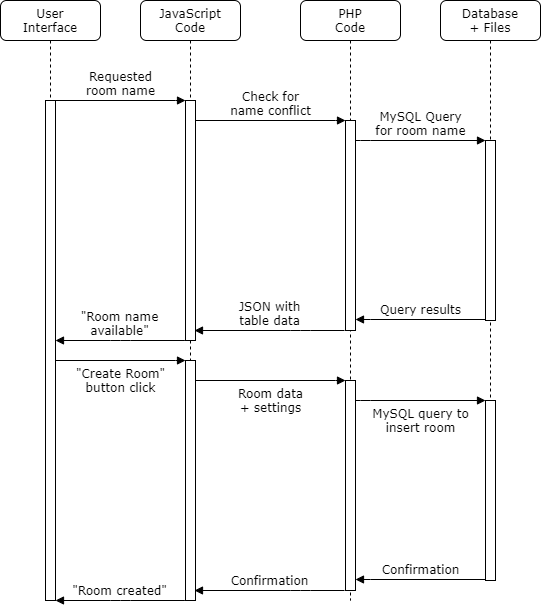
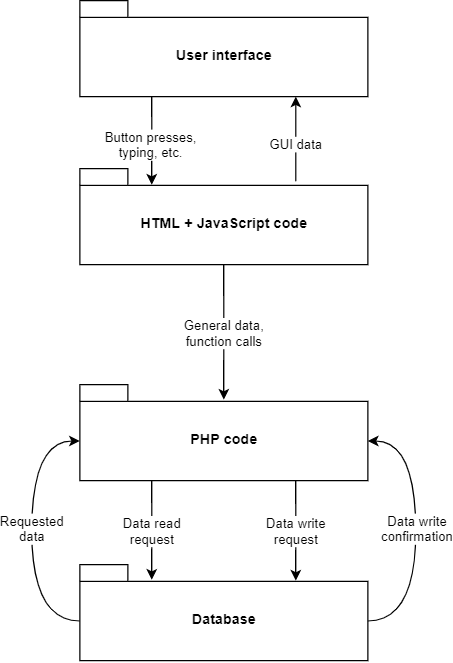
* Brackets, open software, will be used for basic text editing and creating files. License: <https://github.com/adobe/brackets/blob/master/LICENSE>
* Sublime text, open software, will be used for basic text editing and creating files. License: See page 7
* Bootstrap, a premade extension for CSS styling, will be used as a foundation for the site’s UI. License: <https://github.com/twbs/bootstrap/blob/v4.0.0/LICENSE>
* SourceTree, open software, will be used to more easily access and modify the Git repository used for the project. License: No license is used for this software. License: <https://www.atlassian.com/legal/software-license-agreement>
* Cyberduck, open software, will be used as a portal for accessing FAU’s LAMP server to upload files. License: <https://cyberduck.io/license/>

Site:

* Data will be stored on FAU’s MySQL servers. These servers will be accessed by the site using PHP code, and databases will be created using FAU’s phpMyAdmin.
* The video player in each room will use the YouTube API to display and control content. Terms of Service: <https://developers.google.com/youtube/terms/api-services-terms-of-service>
* The retrieval of video thumbnails, as well as the search function listed in requirement 3a.ii, requires use of the YouTube Data API, which requires a key to use. Key: AIzaSyBiI3RW4tTmXKWV1GyUU8IsLUyfzBHVuAE
* A chat module will be used and adapted to fit our needs. It is free to use and can be found here: <https://coursesweb.net/php-mysql/script-chat-simple_s2>

# High-Level UML Diagrams

System Architecture Diagram (left) and Sequence Diagram (right):



Due to the nature of our project, our team has decided that an object-oriented approach is not suitable for implementation. Thus, a class diagram would not be informative, and is therefore not shown here.

Additionally, note that the interaction described in the sequence diagram above is an example, and may not reflect an actual interaction in the end product (though the structure of the process will be implemented).

# Project Risks

Risks involved in developing the project, ordered by combined severity and probability

1. Skill Risks: Our team is somewhat inexperienced with software and web development, especially in a group setting. Certain requirements (such as functional requirement 2d.ii) have been identified that may pose an issue during development.   
   The probability of this risk posing a problem is rather high, and the severity of a possible problem would be moderate (large features go unimplemented).
2. Schedule Risks: Given our 2-4 week development schedule, we may not be able to sufficiently implement all functional and non-functional requirements in time. The team has little experience with group software development, making it difficult to gauge the development time for various functionalities.   
   The probability of this risk posing a problem is unidentified, and the severity of a possible problem could range from minor to extremely limiting, depending on what requirements are prioritized.
3. Technical Risks: After investigation of requirements, all site functions have been determined to be possible to implement, though possibly difficult (see Skill Risks). No single site feature is completely original to Cinemus, meaning that all features are theoretically possible. However, there is a low chance that restrictions such as server capacity and tool access may pose an inescapable problem with certain requirements.  
   The probability of this risk posing a problem is extremely low and well-researched, but the severity of a possible problem would be major (leaving out entire services and functionalities)
4. Teamwork Risks: Of the four team members, one has long periods of inactivity before returning to help with the project. As such, the number of members developing the project at any given time is often hard to predict; however, the loss of a team member is not entirely debilitating.  
   The probability of this risk posing a problem is unknown but somewhat high, and the severity of a possible problem would be minor (increasing the probability of Skill and Schedule Risks).
5. Legal Risks: Two major risks of this type have been identified. First, Cinemus would, by design, use and show videos from YouTube. However, our method of accessing these videos (the YouTube Player API) is open for the public to use freely, so this is unlikely to pose a problem. Second, current Cinemus designs look similar to the streaming platform Twitch, and the main concept is rather similar. These problems are addressed in Competitive Analysis.  
   The probability of this risk posing a problem is virtually zero, but any possible problem would completely stop the project, as it would conflict with the core site idea.

# Team

Trello Board: <https://trello.com/b/87Z9slrr/cen4010-group-project>

Git repository: <https://github.com/cen4010-s2020-g12/Group-12-Project/tree/milestone-1>

Group Members (ordered alphabetically by last name):

* Clifford Ariel
  + Role: Front-End Developer
  + Created unfinished “YouTube API” section of vertical prototype
  + Helped review M3 deliverable
  + Created YouTube presentation of vertical prototype
  + 22/100 points
* Preston Billion Polak
  + Role: Scrum Master, Back-End Developer
  + Created M3 deliverable
  + Created framework and Database section of vertical prototype
  + Created final UML diagrams
  + Discussed implementation details
  + Organized team near end of M3 development cycle
  + 26/100 points
* Jorge Risco
  + Role: Front-End Lead
  + Created initial UML diagram drafts
  + Heavily reviewed M3 deliverable
  + Discussed and helped document chat feature
  + Discussed and theorized implementation details
  + 26/100 points
* Madison Verger
  + Role: Back-End Lead
  + Reviewed M3 deliverable
  + Heavily researched and documented chat feature
  + Created Chat section of vertical prototype
  + Discussed and theorized implementation details
  + Took lead near beginning of M3 development cycle
  + 26/100 points

The vertical prototype can be found at <https://lamp.cse.fau.edu/~cen4010s2020_g12/Project/VerticalPrototype/test.php>.

Sublime license:

----- BEGIN LICENSE ----- Madison G Verger Single User License EA7E-1164024 AD05A4F6 273333FD CBC61231 E8503D0A 88ECE20E 3B17B0C7 2D163CB2 ED68D2C9 22FB02AA E09C7216 0BE9C4AF 384D9E85 EAE8235C 5B55E68E 368B1353 9ED22F6A 328E9E3B 1444941A 258A0833 F5C6C445 62394299 553A22C4 6D3E3E48 8BCEF3F2 7F7737A7 40B913DA 120B30A3 DD3208E6 3E743289 5E7B1F8C 42257F46 10CA8CD5 ------ END LICENSE ----