Software Engineering Semester Project
Video Streaming Platform: Amaranth

Submitted by:
Hassan Ismail i22-1011
Rehal Saeed: i22-1071
Samra Mashaam: i22-0757

Project Introduction:

Amaranth is a video streaming service available for free for users of all ages. This platform removes the issue of irritating mid-roll ads that exist in nearly every other video streaming platform. This also addresses other issues, such as censorship of content to appease advertisers. This means that content creators often find their content demonetised, restricted, or removed entirely from the platform, affecting their income and viewership. This also ruins the experience for viewers and silences discussion on important topics.

Due to the removal of midroll ads, Amaranth will also empower content creators by providing them with alternate revenue streams including brand deals, subscriptions, and donations through their viewers. With the help of these technologies, creators will be able to make money directly from the platform and their viewers, without relying on sponsors, which will give them more autonomy over the financial success of their work. By implementing these programs, Amaranth hopes to guarantee everyone's safety and happiness while fostering a more balanced and creator-friendly environment.

Functional Requirements:

1. User Registration: The system shall allow users to register themselves as viewers or content creators, with appropriate permissions.
2. Video search: The system shall allow users to search for videos and it will present the appropriate results.
3. Video viewing: The system shall support pausing, playing and skipping through videos.
4. Video engagement: The system shall allow users to like, dislike, flag, and leave comments on videos.
5. Content creation: The system shall allow registered content creators to upload videos.
6. Subscriptions and donations: The system shall allow users to subscribe to creators for exclusive content, and donate money to show support.
7. Following creators: The system shall allow users to follow their preferred content creators.
8. Kids mode registration: The system shall allow users to create a kid-friendly account with restricted content access.
9. Platform compliance: The system shall allow content creators to be able to disclose sponsorships and mark content rating as kid-friendly or not.
10. Account settings: The system shall allow users to set their account settings.

Non-Functional Requirements:

Product Requirements:

1. The system shall be able to handle at least 10,000 users without noticeable lag.
2. The system shall return search results within 1 second.
3. The system shall have an interface that is easy to use and accessible.
4. The system shall be backed up and recover from crashes within five minutes.

5. The system shall protect user data and transactions.
6. The system shall only allow users to have permissions of their own accounts (i.e viewers, and content creators).

Organizational Requirements:

1. The developers shall use agile methodology with regular sprints and meetings.
2. The developers shall ensure that testing must be conducted before every release.
3. The developers shall use Github and Trello to maintain collaboration and communication.

External Requirements:

1. The system must comply with local laws regarding the protection of user data.
2. The system must comply with local laws regarding the content and privacy in kids mode.
3. The system must comply with laws and regulations regarding financial transactions.
4. The system can use external payment methods for financial transactions.

User stories:

1. As a viewer, I want to search for a topic and see matching results so that I can pick a video to watch.

   Acceptance Criteria:

   Given the viewer is on the platform's search page,
   When they enter a search query and submit it,
   Then the system should return a list of videos relevant to the query.
   And the results should be sorted by relevance by default.

2. As a viewer, I want to leave a like or dislike on a video so that I can quickly show my thoughts on it.

   Acceptance Criteria:

   Given the viewer is watching a video,
   When they click the like or dislike button,
   Then the system should register the reaction.
   And the system should display that change.

3. As a viewer, I want to leave a comment on a video so that I can express my opinion on it.

   Acceptance Criteria:

   Given the viewer is watching a video and is logged in,
   When they enter a comment and submit it,
   Then the system should display the comment under the video.
   And the comment should be visible to other users.

4. As a viewer, I want to follow a content creator so that I can easily find their content again.

   Acceptance Criteria:

   Given the viewer is on a content creator's channel,
   When they click the follow button,
   Then the system should add the creator to the viewer's following list.
   And the viewer should receive updates about the creator's new uploads.

5. As a viewer, I want to access and edit my following list so that I can change it to my preferences.

   Acceptance Criteria:

   Given the viewer is logged in and has followed some creator(s),
   When they go to their following list,
   Then they should see all the creators they are following.
   And they should be able to unfollow or manage notifications for each creator.

6. As a user, I want to register with the platform so that my preferences and activity history can be stored under my name.

   Acceptance Criteria:

   Given a new user wants to register,
   When they provide the required information (e.g., username, email, password),
   Then the system should create an account for them,
   And they should be able to log in with their credentials.

7. As a content creator, I want to register with the platform so that I can upload videos and have a dedicated account under my name.

   Acceptance Criteria:

   Given a user wants to become a content creator,
   When they complete the registration and agree to creator terms,
   Then the system should grant them access to upload videos and manage their channel.

8. As a viewer, I want to be able to transfer money over the platform so that I can donate to my favorite content creators.

Acceptance Criteria:

Given the viewer wants to donate,
When they select a content creator and enter the donation amount,
Then the system should securely process the transaction.
And the creator should receive the donation in their balance.

9. As a user, I want to be able to access my account settings and history so that I can review and edit them to match my preferences.

   Acceptance Criteria:

   Given a user is logged in,
   When they navigate to account settings,
   Then they should be able to view and edit their profile, preferences, and activity history.

10. As a viewer, I want to subscribe to a content creator so that I can access and view their exclusive content.

    Acceptance Criteria:

    Given the viewer is on a content creator's channel,
    When they click the subscribe button and confirm payment,
    Then the system should grant them access to exclusive content from that creator.

11. As a viewer, I want to be able to access and review my subscriptions so that I can cancel or renew them.

    Acceptance Criteria:

    Given a viewer is subscribed to one or more creators,
    When they navigate to the subscription management page,
    Then they should be able to view, cancel, or renew their subscriptions.

12. As a viewer, I want to flag harmful content so that the video and/or the creator can be reviewed.

    Acceptance Criteria:

    Given a viewer is watching a video,
    When they click the report button and select a reason,
    Then the system should submit the report for review by moderators.
    And the system should alert the user that their flag has been acknowledged

13. As a content creator, I want to upload a video so that the viewers can watch it.

    Acceptance Criteria:

    Given a content creator is logged in,
    When they upload a video with necessary details,
    Then the system should process and publish the video.

14. As a viewer, I want to create a Kid's channel so that my child can watch videos safely.

    Acceptance Criteria:

    Given a viewer wants to create a child-friendly environment,
    When they set up a Kid's channel,
    Then only kid-friendly content should be displayed on it.

15. As a viewer, I want to pause, play, and skip through a video so that I can watch it at my pace.

    Acceptance Criteria:

    Given a viewer is watching a video,
    When they use the player controls,
    Then the video should pause, resume, or skip accordingly.

16. As a content creator, I want to make it known that a video has a brand sponsorship in it, so that I can follow platform guidelines.

    Acceptance Criteria:

    Given a content creator is uploading a sponsored video,
    When they mark it as containing a sponsorship,
    Then the system should display a sponsorship disclosure.

17. As a content creator, I want to make it known if a video is for kids or adults only, so that I can follow platform guidelines.

    Acceptance Criteria:

    Given a content creator is uploading a video,
    When they select the audience category,
    Then the system should apply the appropriate content restrictions.

Product Backlog:

1. As a viewer, I want to search for a topic and see matching results so that I can pick a video to watch.
Priority: High

2. As a viewer, I want to leave a like or dislike on a video so that I can quickly show my thoughts on it.
Priority: Medium

3. As a viewer, I want to leave a comment on a video so that I can express my opinion on it.
Priority: Medium

4. As a viewer, I want to follow a content creator so that I can easily find their content again.
Priority: Medium

5. As a viewer, I want to access and edit my following list so that I can change it to my preferences.
Priority: Medium

6. As a user, I want to register with the platform so that my preferences and activity history can be stored under my name.
Priority: High

7. As a content creator, I want to register with the platform so that I can upload videos and have a dedicated account under my name.
Priority: High

8. As a viewer, I want to be able to transfer money over the platform so that I can donate to my favorite content creators.
Priority: Medium

9. As a user, I want to be able to access my account settings and history so that I can review and edit them to match my preferences.
Priority: Medium

10. As a viewer, I want to subscribe to a content creator so that I can access and view their exclusive content.
Priority: Medium

11. As a viewer, I want to be able to access and review my subscriptions so that I can cancel or renew them.
Priority: Medium

12. As a viewer, I want to flag harmful content so that the video and/or the creator can be reviewed.
Priority: Low

13. As a content creator, I want to upload a video so that the viewers can watch it.
Priority: High

14. As a viewer, I want to create a Kid's channel so that my child can watch videos safely.
Priority: High

15. As a viewer, I want to pause, play, and skip through a video so that I can watch it at my pace.
Priority: High

16. As a content creator, I want to make it known that a video has a brand sponsorship in it, so that I can follow platform guidelines.
Priority: Medium

17. As a content creator, I want to make it known if a video is for kids or adults only, so that I can follow platform guidelines.
Priority: Medium

Sprint 1 backlog:

6. As a user, I want to register with the platform so that my preferences and activity history can be stored under my name.
Priority: High

7. As a content creator, I want to register with the platform so that I can upload videos and have a dedicated account under my name.
Priority: High

9. As a user, I want to be able to access my account settings and history so that I can review and edit them to match my preferences.
Priority: Medium

13. As a content creator, I want to upload a video so that the viewers can watch it.
Priority: High

14. As a viewer, I want to create a Kid's channel so that my child can watch videos safely.
Priority: High

15. As a viewer, I want to pause, play, and skip through a video so that I can watch it at my pace.
Priority: High

Sprint 2 backlog:

3. As a viewer, I want to leave a comment on a video so that I can express my opinion on it.
Priority: Medium

4. As a viewer, I want to follow a content creator so that I can easily find their content again.
Priority: Medium

4. As a viewer, I want to follow a content creator so that I can easily find their content again.
Priority: Medium

5. As a viewer, I want to access and edit my following list so that I can change it to my preferences.
Priority: Medium

10. As a viewer, I want to subscribe to a content creator so that I can access and view their exclusive content.
Priority: Medium

11. As a viewer, I want to be able to access and review my subscriptions so that I can cancel or renew them.
Priority: Medium

Sprint 3 backlog:

1. As a viewer, I want to search for a topic and see matching results so that I can pick a video to watch.
Priority: High

2. As a viewer, I want to leave a like or dislike on a video so that I can quickly show my thoughts on it.

Priority: Medium

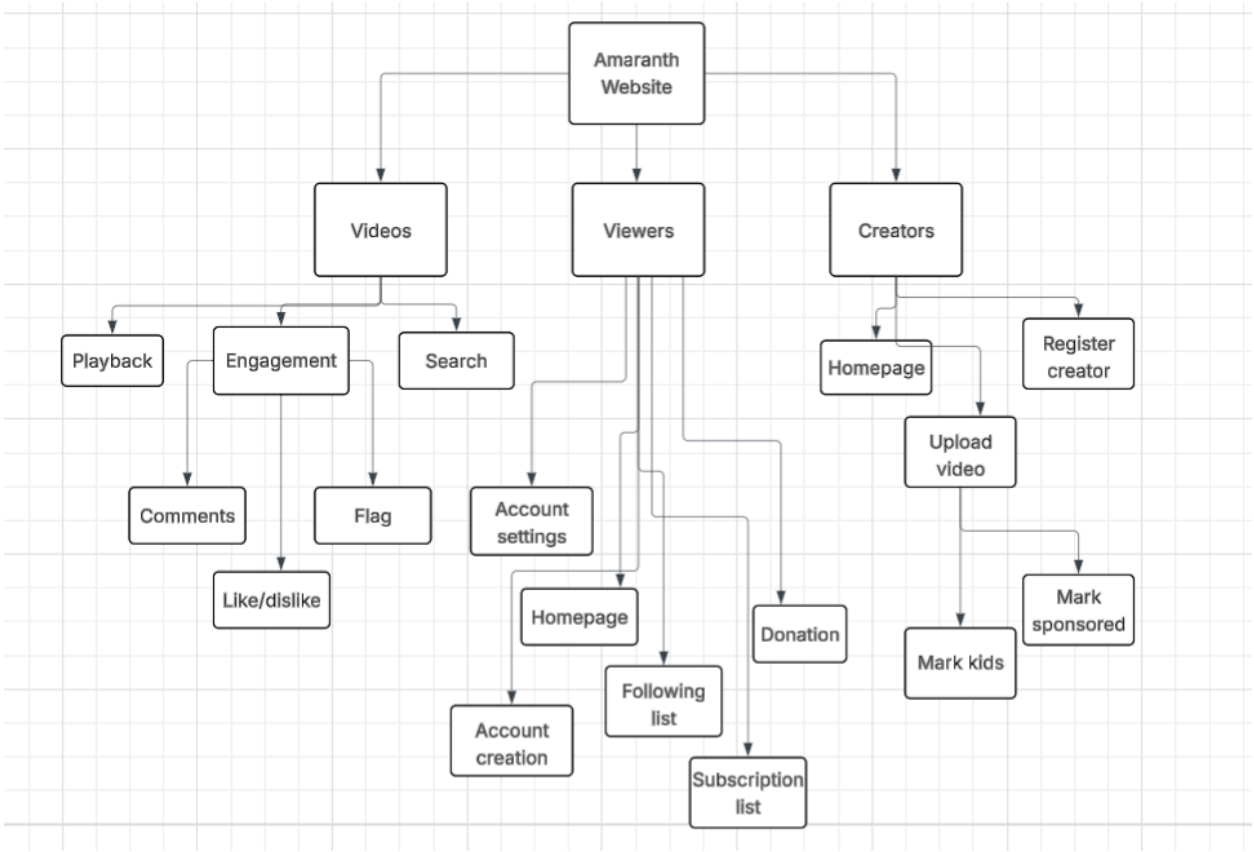
8. As a viewer, I want to be able to transfer money over the platform so that I can donate to my favorite content creators.
Priority: Medium

12. As a viewer, I want to flag harmful content so that the video and/or the creator can be reviewed.
Priority: Low

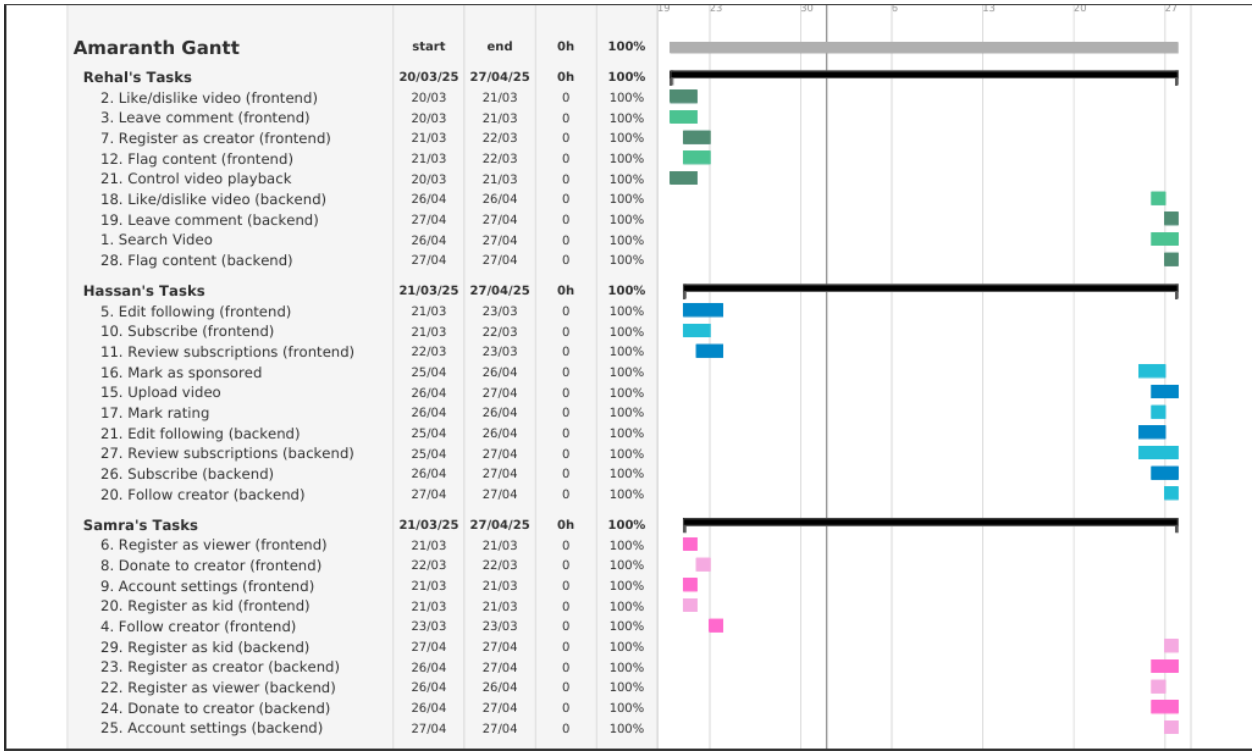
16. As a content creator, I want to make it known that a video has a brand sponsorship in it, so that I can follow platform guidelines.
Priority: Medium

17. As a content creator, I want to make it known if a video is for kids or adults only, so that I can follow platform guidelines.
Priority: Medium

Software Project Plan (Work breakdown):



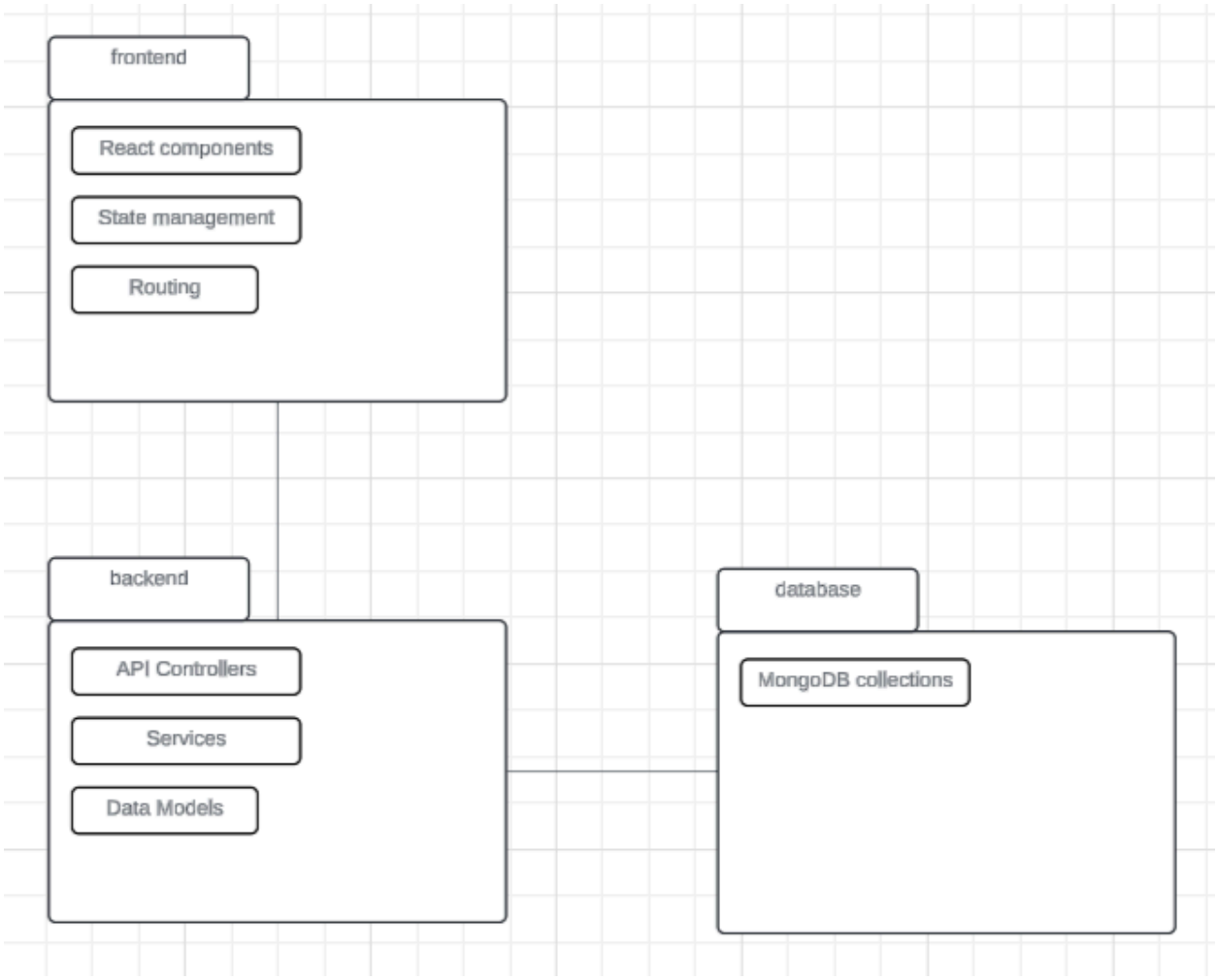
Software Project Plan (Gantt Chart):



Amaranth Gantt	start	end	0h	100%
Rehal's Tasks	**20/03/25**	**27/04/25**	**0h**	**100%**
2. Like/dislike video (frontend)	20/03	21/03	0	100%
3. Leave comment (frontend)	20/03	21/03	0	100%
7. Register as creator (frontend)	21/03	22/03	0	100%
12. Flag content (frontend)	21/03	22/03	0	100%
21. Control video playback	20/03	21/03	0	100%
18. Like/dislike video (backend)	26/04	26/04	0	100%
19. Leave comment (backend)	27/04	27/04	0	100%
1. Search Video	26/04	27/04	0	100%
28. Flag content (backend)	27/04	27/04	0	100%
Hassan's Tasks	**21/03/25**	**27/04/25**	**0h**	**100%**
5. Edit following (frontend)	21/03	23/03	0	100%
10. Subscribe (frontend)	21/03	22/03	0	100%
11. Review subscriptions (frontend)	22/03	23/03	0	100%
16. Mark as sponsored	25/04	26/04	0	100%
15. Upload video	26/04	27/04	0	100%
17. Mark rating	26/04	26/04	0	100%
21. Edit following (backend)	25/04	26/04	0	100%
27. Review subscriptions (backend)	25/04	27/04	0	100%
26. Subscribe (backend)	26/04	27/04	0	100%
20. Follow creator (backend)	27/04	27/04	0	100%
Samra's Tasks	**21/03/25**	**27/04/25**	**0h**	**100%**
6. Register as viewer (frontend)	21/03	21/03	0	100%
8. Donate to creator (frontend)	22/03	22/03	0	100%
9. Account settings (frontend)	21/03	21/03	0	100%
20. Register as kid (frontend)	21/03	21/03	0	100%
4. Follow creator (frontend)	23/03	23/03	0	100%
29. Register as kid (backend)	27/04	27/04	0	100%
23. Register as creator (backend)	26/04	27/04	0	100%
22. Register as viewer (backend)	26/04	26/04	0	100%
24. Donate to creator (backend)	26/04	27/04	0	100%
25. Account settings (backend)	27/04	27/04	0	100%

System Architecture (Subsystems):

The architecture consists of three subsystems: The frontend, database, and the backend operations. The frontend, made with React, presents a clean and visually appealing UI, page routing, state management using hooks, takes user input, and is generally in charge of the client side. The database is made using MongoDB, and stores the data in a well structured manner. The backend subsystem handles the APIs, controllers, and the mongoose models.

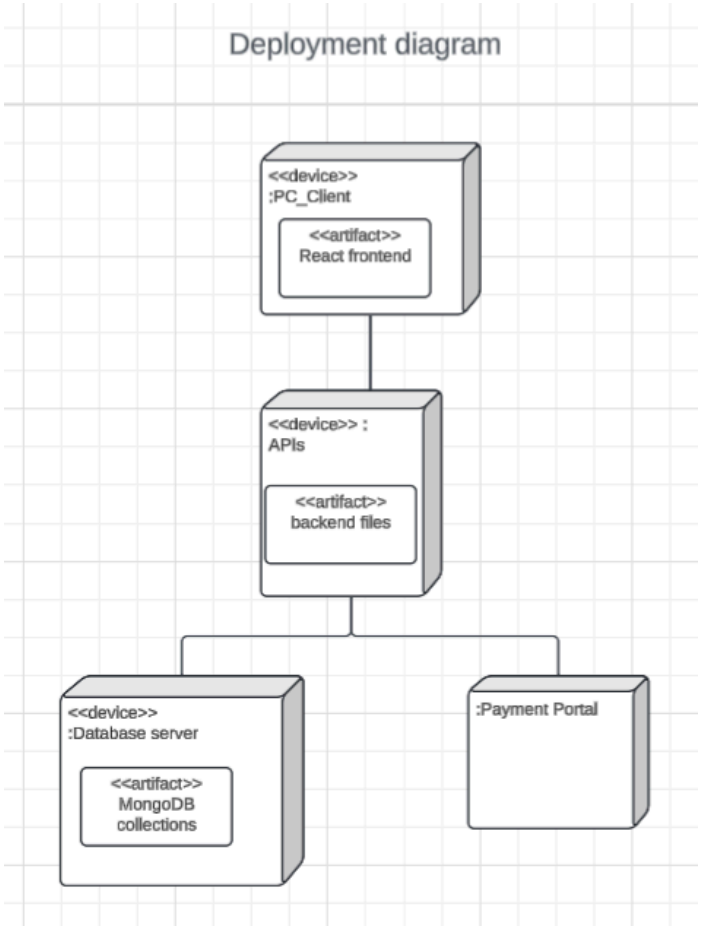


System Architecture (Architecture Styles):

Layered Architecture: The frontend, backend, and database layers are all separated and depend on each other in a layered manner. The frontend cannot connect directly to the database. This allows for better modularity and gives a clear separation of concerns.

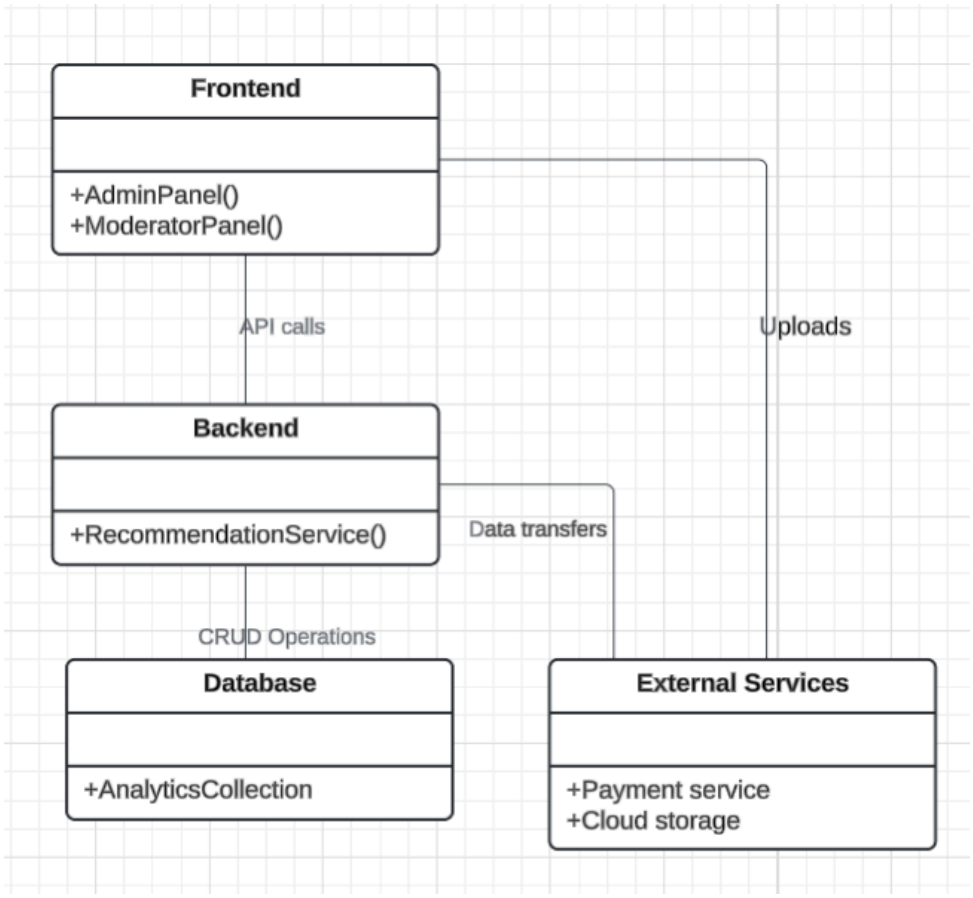
Client-server architecture: The frontend (React) acts as a client and makes requests to the backend (Express) which acts as the server. This works well because the data can be stored centrally and different types of clients can be handled.

System Architecture (Deployment Diagram):

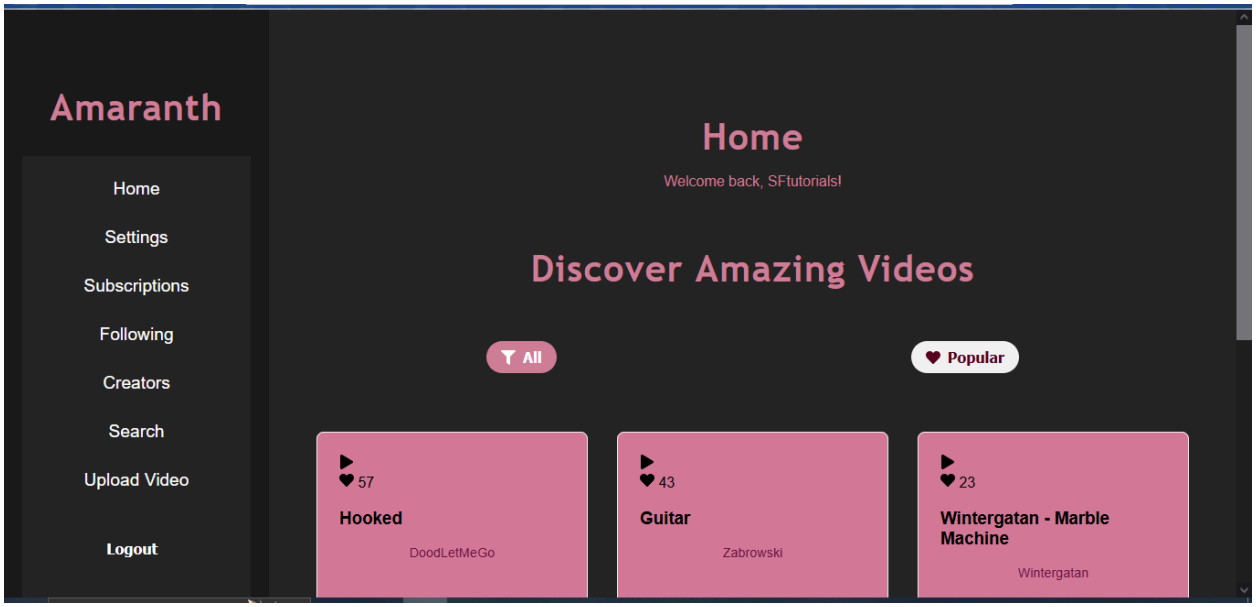


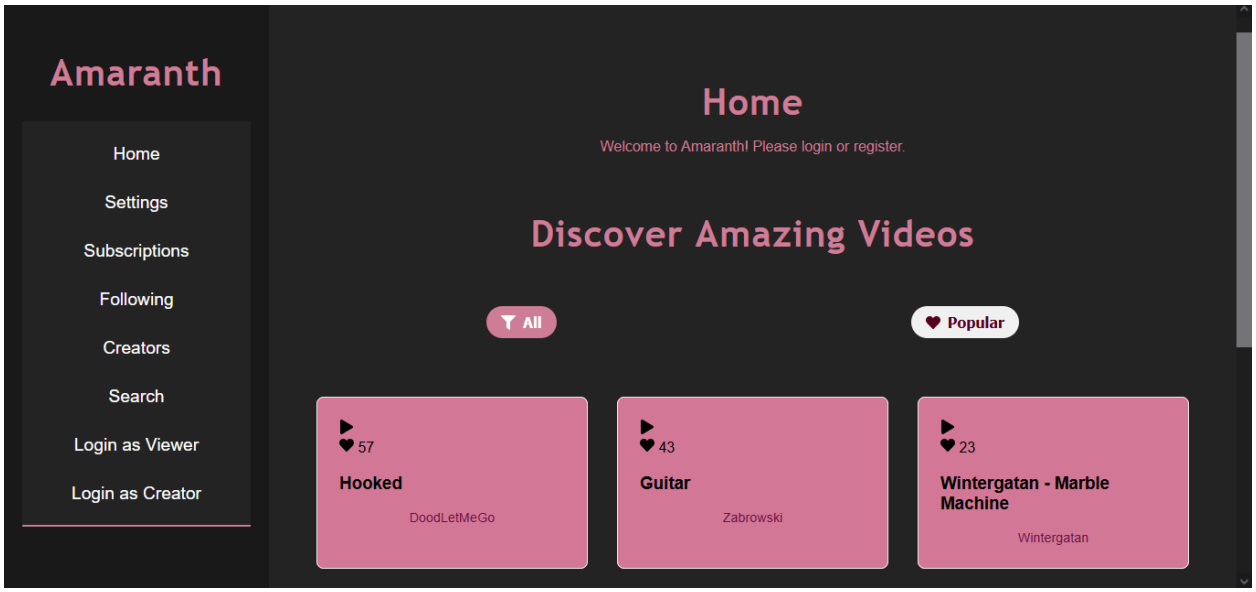
System Architecture (Future Components):

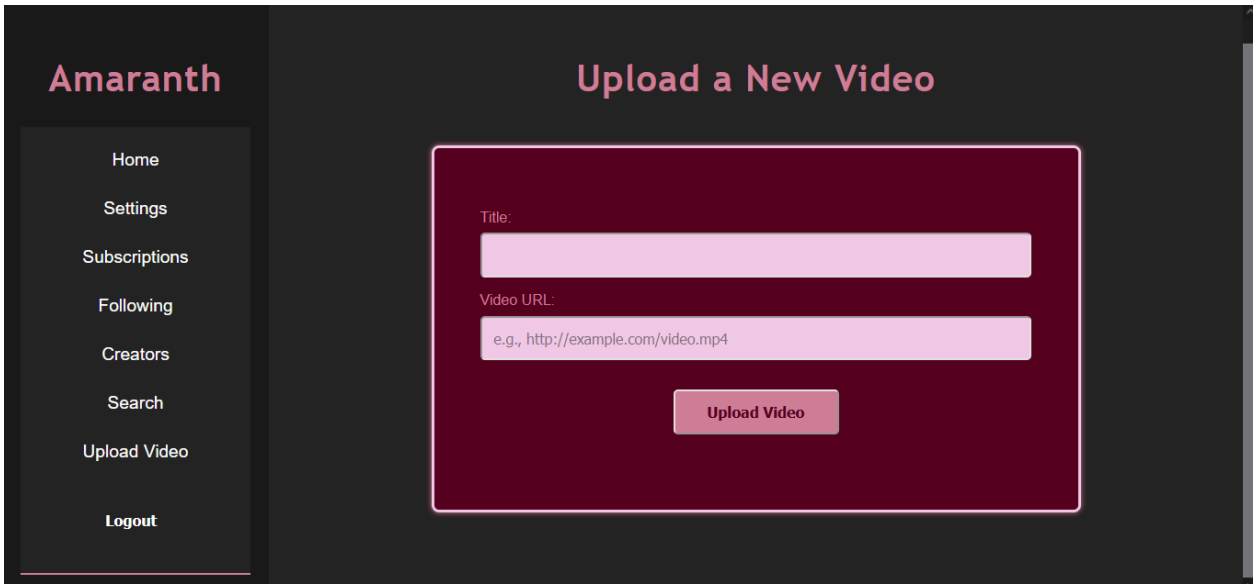
Expected additions for future enhancements:

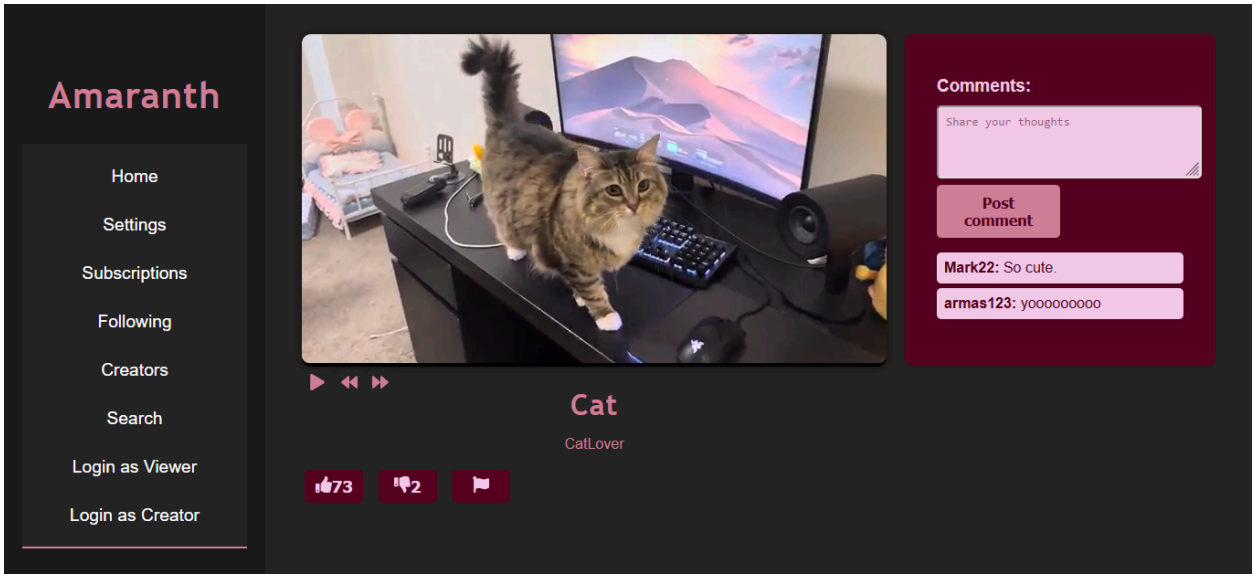


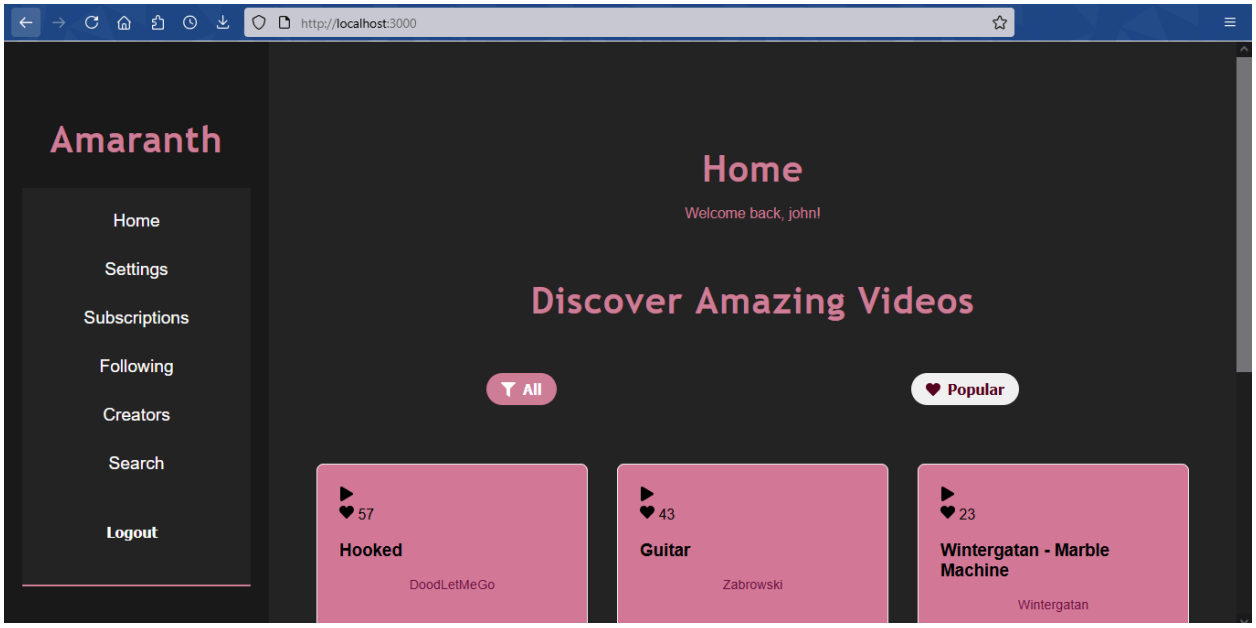
Implementation Screenshots:




Amaranth
Home
Settings
Subscriptions
Following
Creators
Search
Login as Viewer
Login as Creator
Home
Welcome to Amaranth! Please login or register.
Discover Amazing Videos
All
Popular
57
Hooked
DoodLetMeGo
43
Guitar
Zabrowski
23
Wintergatan - Marble Machine
Wintergatan

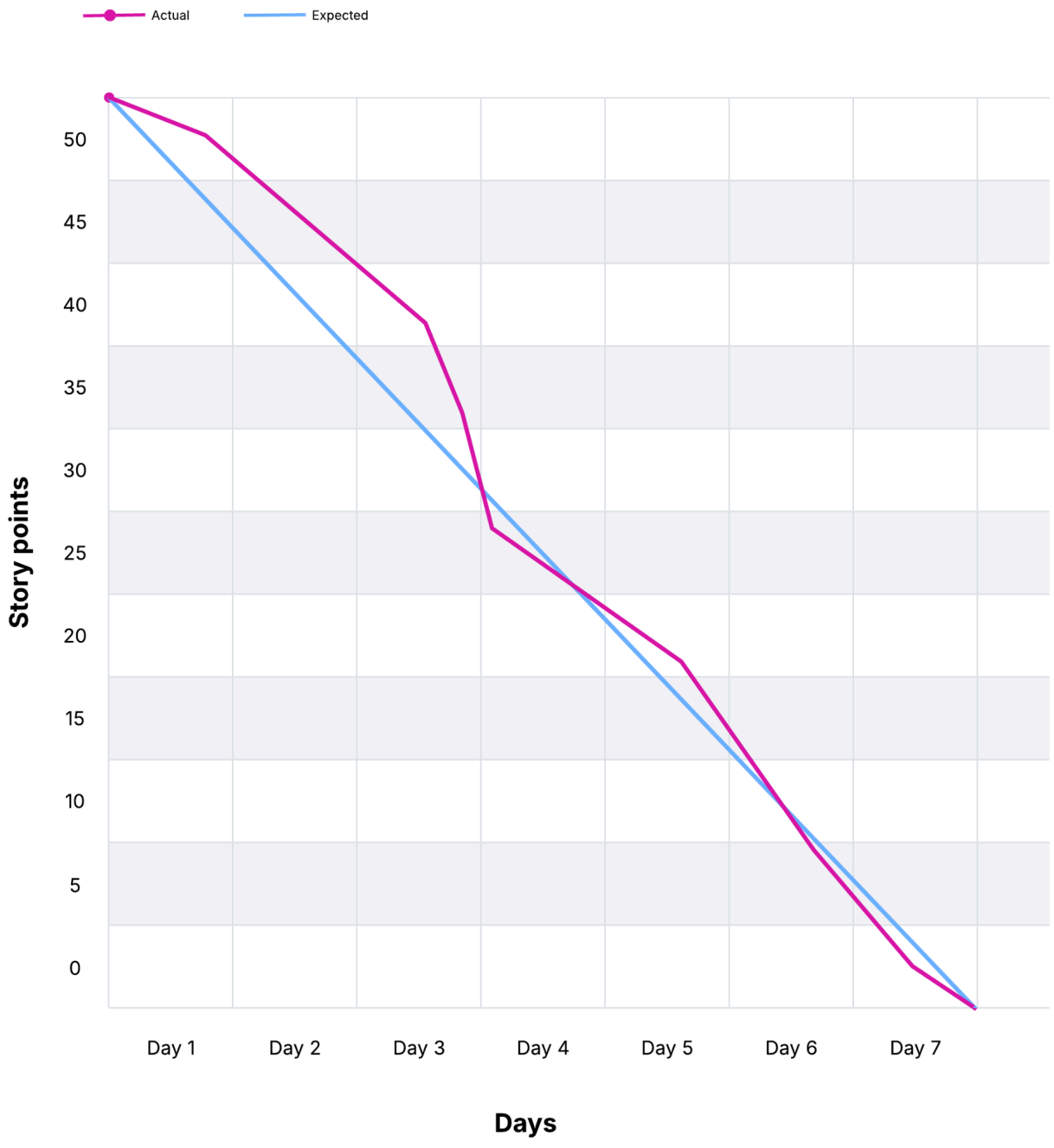

Amaranth
Home
Settings
Subscriptions
Following
Creators
Search
Upload Video
Logout
Upload a New Video
Title:
Video URL:
e.g., http://example.com/video.mp4
Upload Video



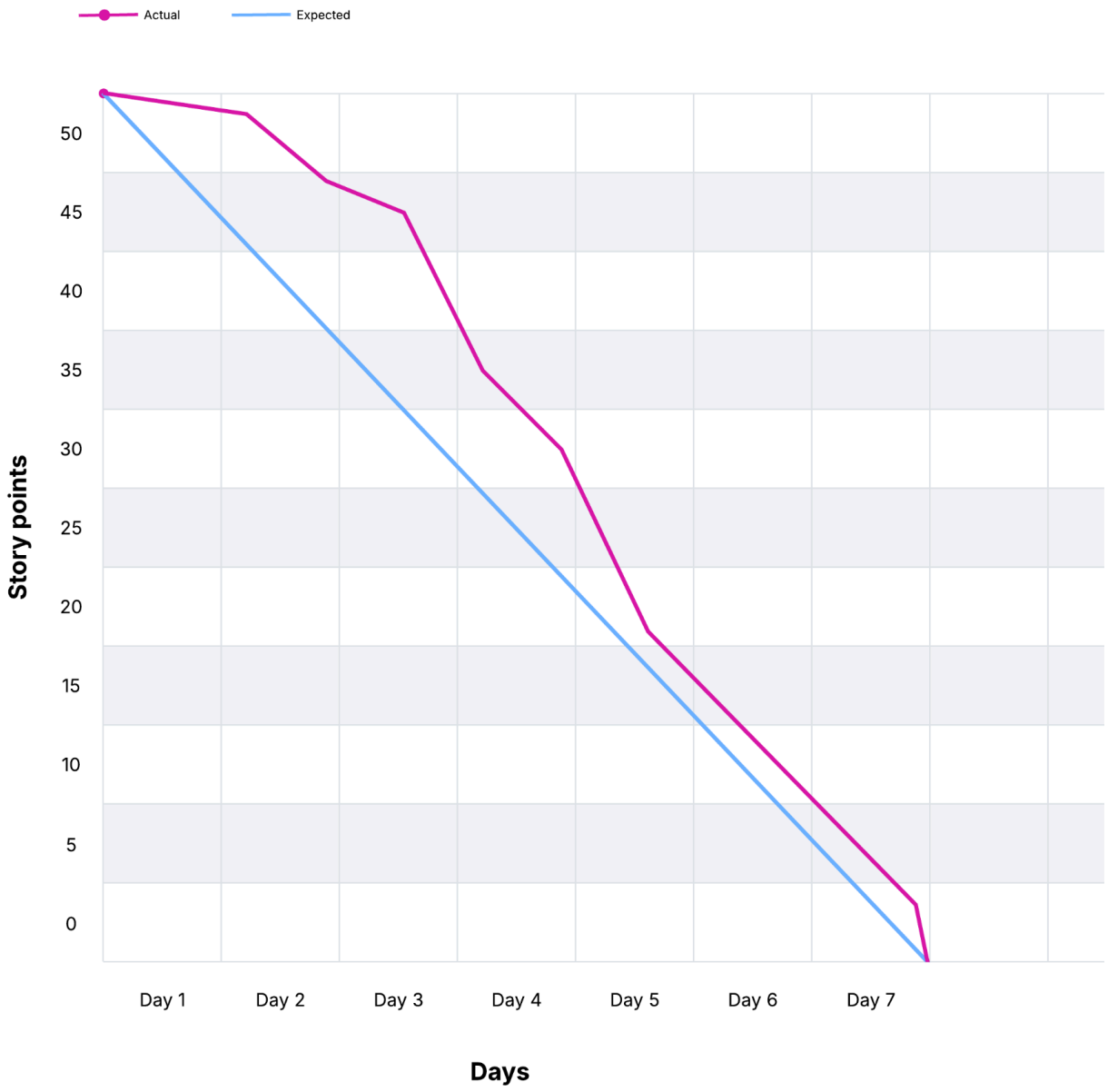


Product Burndown Chart:

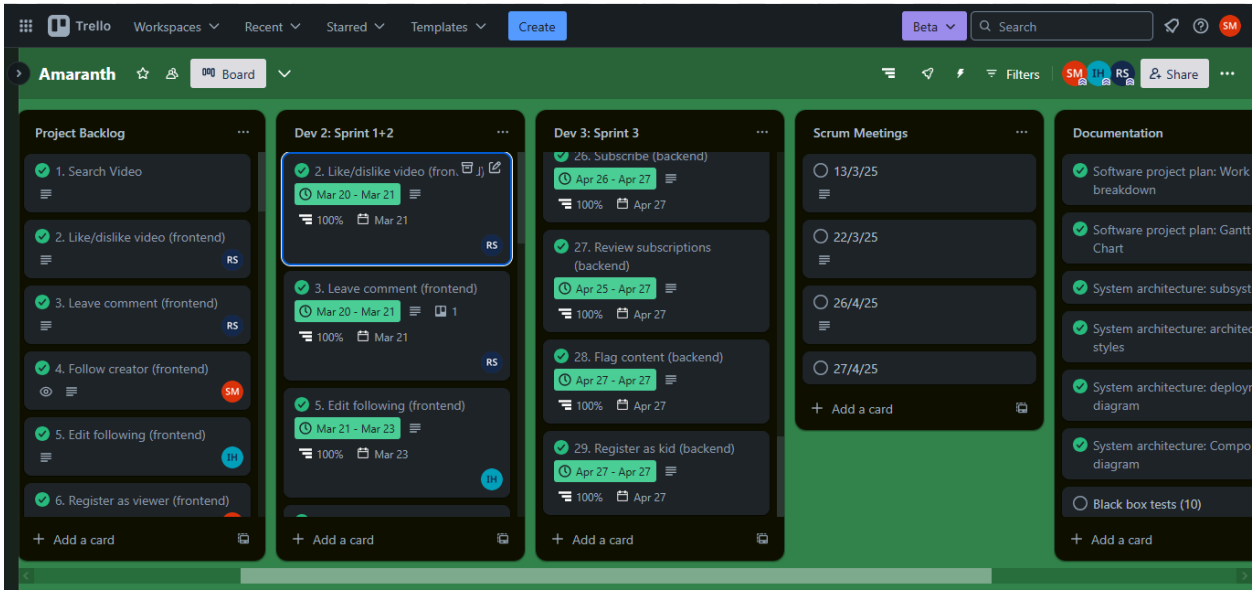
Sprint 1+2:

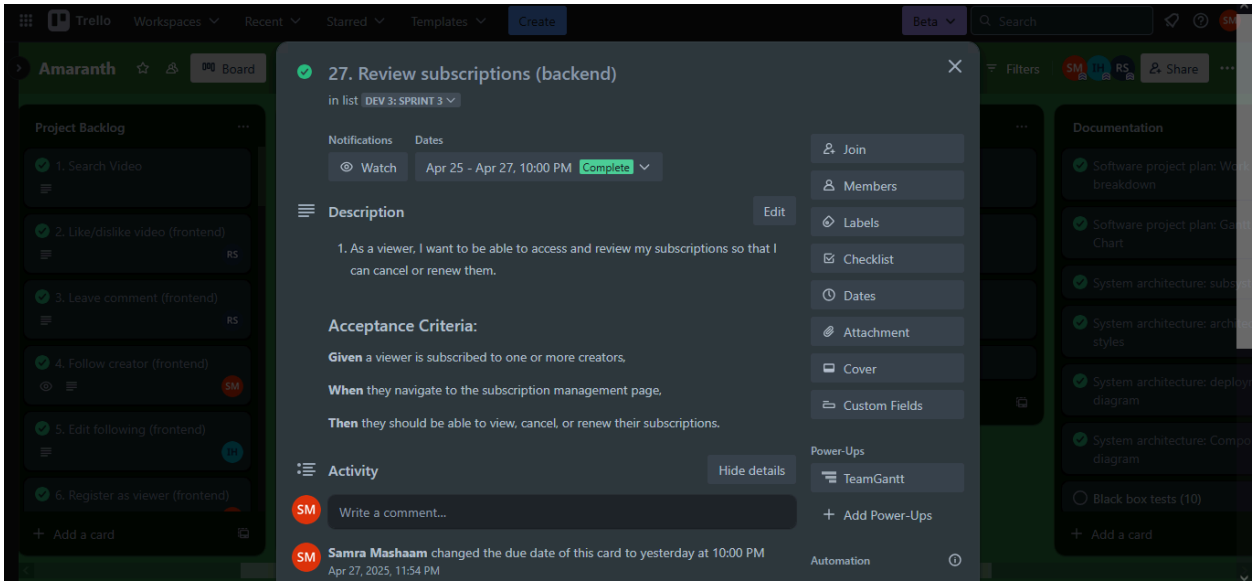

Actual
Expected
Story points
50
45
40
35
30
25
20
15
10
5
0
Day 1
Day 2
Day 3
Day 4
Day 5
Day 6
Day 7
Days

Sprint 3:


Actual
Expected
Story points
50
45
40
35
30
25
20
15
10
5
0
Day 1
Day 2
Day 3
Day 4
Day 5
Day 6
Day 7
Days

Trello Board screenshots:





Black box test cases:

Test case 1: TC_VIDEO_01
Feature: Video player
Priority: High
Test type: Function
Equivalence class: All videos from any creator.
Steps:

1. Navigate to a video page
2. Verify the video starts within a second
3. Click the skip buttons and verify it skips forward and backward by 10 seconds
4. Click the pause/play button and verify the video pauses/plays accordingly

Expected Results:
Video loads with pause, play, and skip buttons.
Skip buttons skip 10 seconds each.
Video pauses and plays when required.

Actual results: Pass
Edge cases: Video of 1 second, video of 1 hour.

Test case 2: TC_LK_02
Feature: Like/dislike system
Test type: Functional
Priority: High
Equivalence class: All registered users like/dislike feedback
Steps:

1. Navigate to a video
2. Click the like (thumbs up) button
3. Refresh page or navigate back to the page
4. Repeat for dislike

Expected results:
Count goes up when clicked
Count goes back down clicked again, or when other button is clicked
Count stays after refresh

Actual results: Pass
Edge cases: Spamming the button is handled without issues, 0 likes are incremented to one, max integer likes

Test case 3: TC_CMT_03
Feature: Comments
Priority: Medium
Test type: Input validation
Equivalence class: All comments from viewer or content creator account on any video
Steps:

1. Navigate to a video page
2. In the comments box, enter this text: “Test characters: 123@!#$”

3. Submit the comment
4. Refresh the page or navigate back to it

Expected results:
Comment immediately appears in the list with username
The sample text looks the same
The comment stays after refresh
Previous comments show up every time

Actual results: Pass
Edge cases: Empty comment does nothing, 500 character comment

Test case 4: TC_SCH_04
Feature: Search
Test type: Function
Priority: High
Equivalence class: All texts entered in search bar
Steps:

1. Navigate to search
2. In the input box, type “cat”.
3. Click on the first result
4. Ensure it directs to the video player page with the correct video

Expected results:
Results appear within one second.
The results match the text in a case insensitive manner.
The results include creator names and the likes count.

Actual results: Pass
Edge cases: No input returns no results, special characters are ignored, no results show appropriate message.

Test case 5: TC_HM_05
Feature: Homepage display
Test type: Function
Priority: High
Equivalence: All video displays for any viewer
Steps:

1. Navigate to homepage
2. Ensure video cards, buttons, and heading “Discover amazing videos” appear
3. Click “popular” button
4. Click a video card and ensure it goes to the video player of the correct video

Expected results:
Clickable cards (at least three) with video title, creator, and like count appear.
“Popular” button filters out videos with higher like counts.
Video cards lead to correct video.
Video cards shift when hovered, and filter buttons change color to reflect state.

Actual results: Pass
Edge cases: No videos in the DB, none of the UI should be affected

Test case 6: TC_RG_06
Feature: Registration
Test type: Function and user input
Priority: High
Equivalence: All users of ages above 13
Steps:

1. Navigate to sign up page
2. Enter username “username123”
3. Enter password “password123”
4. Enter a valid email address, eg: “i221071@nu.edu.pk”
5. Enter bank number “AK3823487972”
6. Enter age > 30
7. Click submit button
8. Verify alert says “Account created successfully”
9. Verify you are navigated to the home page

Expected results:
Alert pops up to verify account creation
Page is routed to homepage

Actual results: Pass
Edge cases: Field is left empty, system prompts user to enter it. Age entered from 0 to 100. Email format incorrect is alerted.

Test case 7: TC_LG_07
Feature: Login
Test type: Authentication and user input
Priority: High
Equivalence class: All viewers registered in database
Steps:

1. Navigate to login page

2. Enter valid, registered username: “armas123”
3. Enter valid password: “password123”
4. Click submit button
5. Verify the page navigates to the homepage

Expected results:
Page is navigated to the homepage. Homepage displays welcome back message with correct username.

Actual results: Pass
Edge cases: Field is left empty, system prompts user to enter it. Incorrect data receives prompt “invalid credentials”.

Test case 08: TC_UP_08
Feature: Upload video
Test type: Function
Priority: High
Equivalence class: All registered creators and any video with a length from 1 second to an hour.
Steps:

1. Navigate to upload video section
2. Enter video title “Hooked”
3. Enter video path: “/videos/Hooked.mp4”
4. Click submit button
5. Verify success message appears

Expected results:
Success message appears and page is navigated back to the creator’s profile

Actual results: Pass
Edge cases: Invalid path, receives error message. Empty fields receive prompts.

Test case 09: TC_DN_09
Feature: Donation
Test type: Function
Priority: Medium
Equivalence class: All three types of donation values
Steps:

1. From creator page, click the donation button
2. Select a donation value: $10
3. Click the submit button
4. Verify the success message appears

Expected results:
Success message appears. User is navigated back to the creator's page.

Actual results: Pass
Edge cases: No button selected gives a donation of zero. Spamming choice buttons only selects one.

Test case 10: TC_FL_10
Feature: Follow creator
Test type: Function
Priority: Medium
Equivalence class: All registered creators who a user is following
Steps:

1. Navigate to the "following page"
2. Verify a list of creators appears.
3. Click on the "unfollow" button
4. Refresh the page and verify the unfollowed state persists

Expected results: "Unfollow" button changes to reflect "follow" state. Refreshing the page preserves this state.

Actual results: Pass
Edge cases: No followed creators gives an empty page. Spamming button is handled gracefully.

White box test cases:

Test Case ID: WB-VIDEO-001
Component: GET /search route
Objective: Ensure the API returns a 400 error if no search query (q) is provided.
Input Data: No query parameter, just /api/videos/search
Output: HTTP 400, { message: "Search query is required" }
Status: Pass

Test Case ID: WB-VIDEO-002
Component: GET/:id route
Objective: Verify that a valid video ID returns the correct video data.

Input Data: /api/videos/653c1ab7f2d2a9a89f10b2c3
Output: HTTP 200, JSON object with video details.
Status: Pass

Test Case ID: WB-VIDEO-003
Component: GET/:id route
Objective: Verify that an invalid video ID returns error code
Input Data: /api/videos/thisIsNotAnId
Output: HTTP 404, {message: video not found}
Status: Pass

Test Case ID: WB-VIEWER-001
Component: registerViewer(req, res)
Objective: Ensure registration fails if any required field is missing.
Input Data: { username: "john", password: "", email: "john@example.com", bankNum: "1234", age: 25 }
Output: HTTP 400, {message: All fields are required}
Status: Pass

Test Case ID: WB-VIEWER-002
Component: registerViewer(req, res)
Objective: Ensure duplicate usernames are not allowed
Input Data: { username: "john", password: "12345", email: "john@example.com", bankNum: "1234", age: 25 } (john@example.com already exists)
Output: HTTP 400, {message: Username or email already exists}
Status: Pass

Test Case ID: WB-VIEWER-003
Component: loginViewer(req, res)
Objective: Ensure login fails if username or password is missing
Input Data: { username: "john", password: "" }
Output: HTTP 400, {message: Username and password required}
Status: Pass

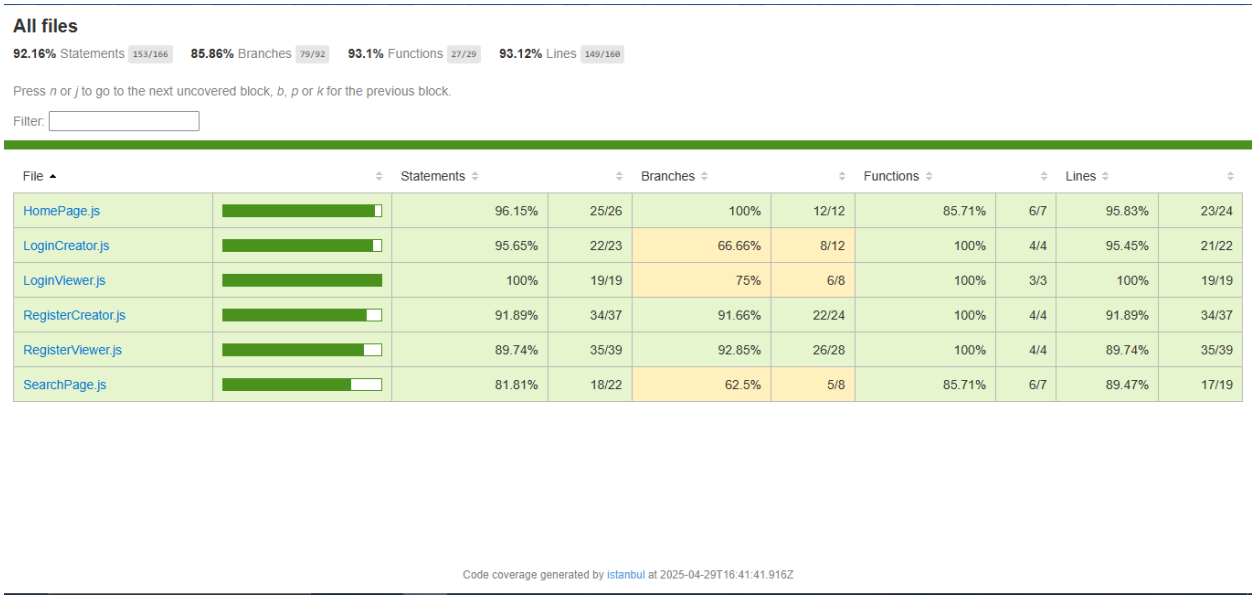
Test Case ID: WB-CREATOR-001
Component: registerCreator(req, res)
Objective: Ensure registration fails when any required field is missing
Input Data: { username: "creator1", password: "", email: "c1@example.com", bankNum: "9876" }
Output: HTTP 400, {message: All fields required}

Status: Pass

Test Case ID: WB-CREATOR-002
Component: registerCreator(req, res)
Objective: Successfully register a new creator when all fields are valid and unique.
Input Data: { username: "creator1", password: "hahahahaha", email: "c1@example.com", bankNum: "9876" }
Output: HTTP 201, {message: creator registered successfully}
Status: Pass

Test Case ID: WB-CREATOR-003
Component: loginCreator(req, res)
Objective: Ensure login fails if username or password is missing
Input Data: { username: "creator1", password: "" }
Output: HTTP 400, { message: "Username and password are required" }
Status: Pass

Test Case ID: WB-CREATOR-004
Component: loginCreator(req, res)
Objective: Fail login when the username does not exist in the database or password doesn't match
Input Data: { username: "invalidUser", password: "none" }
Output: HTTP 400, { message: "Invalid Credentials" }
Status: Pass



All files

92.16% Statements 153/166 85.86% Branches 79/92 93.1% Functions 27/29 93.12% Lines 149/160

Press n or j to go to the next uncovered block, b, p or k for the previous block.

Filter:

File	Statements		Branches		Functions		Lines	
HomePage.js	96.15%	25/26	100%	12/12	85.71%	6/7	95.83%	23/24
LoginCreator.js	95.65%	22/23	66.66%	8/12	100%	4/4	95.45%	21/22
LoginViewer.js	100%	19/19	75%	6/8	100%	3/3	100%	19/19
RegisterCreator.js	91.89%	34/37	91.66%	22/24	100%	4/4	91.89%	34/37
RegisterViewer.js	89.74%	35/39	92.85%	26/28	100%	4/4	89.74%	35/39
SearchPage.js	81.81%	18/22	62.5%	5/8	85.71%	6/7	89.47%	17/19

Code coverage generated by istanbul at 2025-04-29T16:41:41.916Z

Work Division:

Work was divided equally among all members of the team. The general assignment of tasks was done by loosely placing them into groups. While there is slight overlap among the task groups, here is the task division:

Hassan Ismail: Viewing and handling subscriptions and following, and uploading video.
Rehal Saeed: Search video, video playback and controls, video feedback, viewer home page.
Samra Mashaam: Registration pages, login and authentication, donations and account settings.

Lessons Learnt:

The goal of this project was to learn from our past mistakes that occurred in a previous project that went poorly. Here are some of the main things we learned from this experience:

Understanding feasibility: Our previous project was, put simply, not feasible for our small team and limited time. We did not understand the importance of checking feasibility, and were not able to handle the workload in time.

Understanding scope: In a similar way, our scope was too large and there were too many things to handle. This time, we shrunk down the scope of our project considerably.

Understanding our framework: Previously, we jumped straight into the project without any knowledge or understanding of the language or tools we were using. This time, we used a framework we were more familiar with.

Tracking progress: This time, through sprints, burndown charts, and other similar tools, we were able to clearly visualize and understand what was done and what needed to be done, leading to no unexpected surprises at the last minute.

The importance of communication: One of the shortcomings we encountered was that we were not able to have enough scrum meetings, which occasionally led to miscommunication and confusion regarding work.

Properly structured testing: Through black box and white box testing, we were able to get a better understanding of what was working and what needed to be fixed in the project.