**CS673 Software Engineering** 

**Team 2 - Project Name**

**Project Proposal and Planning**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Siddhesh Dighe | Team Leader | *Siddhesh* | 09/28/2023 |
| Pushkar | Design Implementation Leader | *Pushkar* | 09/28 /2023 |
| Sai Ramya Devineni | QA Leader | *Ramya* | 09/28/2023 |
| Ayush Bhaliya | Security Leader | *Ayush* | 09/28/2023 |
| Paridhi Talwar | Requirement Leader | *Paridhi* | 09/28/2023 |
| Deepali Chawla | Configuration Leader | *Deepali* | 09/27/2023 |
| Supriya | Backup QA Leader | *Supriya* | 09/28/2023 |
|  |  |  |  |
|  |  |  |  |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

[Overview](#_g6igqliy7rm)

[Related Work](#_bf21eadgjj29)

[Proposed High level Requirements](#_rgyo4hi9stmq)

[Management Plan](#_ts358bsdtbcv)

[Objectives and Priorities](#_nxeeppkjxgn4)

[Risk Management (need to be updated constantly)](#_tk7yixobah8p)

[Timeline (need to be updated at the end of each iteration)](#_iksrndohvx29)

[Configuration Management Plan](#_j5uvivmxqcsp)

[Tools](#_dzly5b9kz982)

[Deployment Plan if applicable](#_sd8zu6r3jisd)

[Quality Assurance Plan](#_vra5ptwu59qx)

[Metrics](#_vwjduhc9wuah)

[Code Review Process](#_hx3eaiwb8v3m)

[Testing](#_l9xnpmd6hh0y)

[Defect Management](#_5amsh8h9f0c7)

[References](#_pd9euov6m4du)

[Glossary](#_ty3i2nqffhtc)

# Overview

# The use of social media and being constantly bombarded with notifications have made us addicted to our smartphones. This addiction is even more prevalent among young people and students. Platforms like Instagram, TikTok, and Twitter are designed with the goal of keeping us hooked to our phones by providing quick, flashy content that doesn’t require much focus. This makes it difficult to concentrate on one thing that takes more than a few minutes[1]. The consequences are not just students receiving low grades but also students struggling with deep thinking and solving complex problems in different subjects. Video platforms like YouTube provide a lot of educational content but also many distractions that can divert your focus from learning[2]. Solving this problem is where the motivation for our project, *Focused Study* comes from.

Focused Study aims to address the problem of distractions and declining attention spans when it comes to learning. It provides

- **Distraction-Free Learning:** Users can search for educational videos using a single search bar similar to a Google search. The application will then curate a list of the best videos related to the user’s query using YouTube API. Each video would be selected based on multiple factors like best comments, likes-dislikes ratio, and overall views. This helps ensure all the videos suggested on the platform are high-quality.

- **Gamification:** Focused Study helps gamify the process of learning by rewarding users points for staying focused on learning while avoiding distractions. These points can be then used to access more entertainment-focused videos that are aligned with the user’s hobbies and interests.

- **Analytics Dashboard:** While the user is focused on learning from watching curated videos on the platform, behind the scenes the platform tracks the user’s watch time for educational and entertainment videos, to provide a dashboard that gives the user insight into his/her attention span and learning habits

Focused Study uses the following tech stack:

- **Front-end:** React

- **Back-end:** Flask

- **Database:** Firebase Realtime Database + Firestore

- **Deployment:** Docker

# Related Work

- **Forrest App:** Forrest is a timer app that helps you stay focused on the task at hand. It uses the Pomodoro technique to help you stay focused[3].  
  
 **Limitations in Forest App:**

o Forrest app is more focused on tracking and analytics and doesn’t provide the content to learn.

o It’s iOS exclusive and can’t be accessible on Android phones, computers, or on the web

**How is Focused Study different from the Forrest app?**

o Focused Study aims to provide the content required to learn new things while also providing tracking and analytics features similar to the forrest app.

o Since Focused Study is a web app it can be accessible from any device as long as it has a web browser

- **Educational Video Platforms like Udemy, Udacity:** Video platforms like Udacity, Udemy, unlike YouTube, focus only on learning content through courses. They are distraction-free as the only content on the platform is educational courses

**Limitations in Educational Video Platforms:**

o Educational video platforms like Udacity and Udemy are not free and they require you to purchase a course to view these educational videos.

o They only focus on providing you with educational content, while they lack in providing analytics and tracking tools to help users visualize their attention span and learning habits

**How is Focused Study different from other Educational Video Platforms?**

o Since Focused Study will provide educational videos from YouTube, the content on the platform will be freely accessible

o It provides more insights into users’ learning patterns and behaviors through analytics

# Proposed High level Requirements

* 1. Functional Requirements
     1. Essential Features   
          
        Feature Title: Search for videos

Description: As a user, I want to search for topics to learn using a single search bar, so that I can have a distraction-free experience in studying

Estimated hours: 20hrs

Feature Title: Video-watching experience

Description: As a user, I want to watch educational videos without any distractions like comments, recommendations, or related videos so that I can stay focused on the learning experience helping me concentrate on the content

Estimated hours: 5hrs

Feature Title: Analytics Dashboard

Description: As a user, I want to see a dashboard that shows me my watch time for both educational and entertainment videos. It should also show me my attention span while learning so that I can have more insights into my learning habits

Estimated hours: 30hrs

* + 1. Desirable Features:  
         
       Feature Title: Authentication and Profile Management

Description: As a user, I want to log in to the Focused Study web app using my Google account. I also want the ability to manage my profile name, hobbies, and interests, so that I can customize my experience while using the app.

Estimated hours: 10hrs

Feature Title: Entertainment feed to redeem user points

Description: As a user, I want to redeem my user points so that I can watch more entertainment based videos more aligned to my interests and hobbies

Estimated hours: 30hrs

Feature Title: Earn points by staying focused

Description: As a user, I want to earn points for staying focused while learning a topic, so that I can later use these points to watch more entertainment-focused videos that align with my hobbies and interests

Estimated hours: 30hrs

* + 1. Optional Features:  
         
       Feature Title: Share progress with friends

Description: As a user, I want the option to share a snapshot of my dashboard with my friends so that I can motivate them to use the platform for learning.

Estimated hours: 5hrs

Feature Title: Mobile friendly

Description: As a user, I want to access the web app on different mobile platforms like iPhone and Android, so that I can learn seamlessly across different platforms.

Estimated hours: 2hrs

* 1. Nonfunctional Requirements
     1. Security requirements

- User information should be securely stored and not only be accessible to the user that it belongs to.

- The platform should implement proper input validation to mitigate the risk of Cross-Site Scripting (XSS) attacks

- APIs on the platform should only be accessible after proper authentication and authorization

- User activities and system events should be properly monitored in logs on the server

- Third-party libraries should be reviewed and updated often

* + 1. Performance Requirements

- The platform should be smooth and seamless when it comes to response time during user interactions

- Video loading should be quick to minimize user wait time and engagement

- The platform should be able to handle multiple users at the same time without any lags or issues

# Management Plan

## Objectives and Priorities

With this project, our objectives are as follows:

* Completion of all proposed essential features with an intuitive design and flow between the features.
* Successfully containerising the final iteration of the application and ensuring smooth functioning post-deployment.
* Resolving all known bugs in the essential features of the application.
* Following coding standards throughout the iterative development of the application
* Maintaining the application at a high-quality level measured through QA metrics.

## Risk Management (need to be updated constantly)

Our team identified the possible scenarios where we could encounter risks and evaluated their impact on our project using the metrics given. We were able to identify a detailed plan to encounter the risks in the case they do arise.

During iteration 0, we did encounter the following risks and we were able to execute the plan we had decided on:

* Personnel: A new team member was added to the team over this weekend. We already finished lab 1 and started splitting tasks for our iteration 0. So instructions were given to help the new member complete lab 1 and she will be up to date with the team from iteration 1 onwards.
* Technology Competence: All team members are brushing up on concepts and getting doubts clarified related to Git, Flask or React. This is a potential risk but we are working towards minimising the impact on our project.

**Risk Management Sheet Link:** [CS673\_ Team2\_SPPP\_RiskManagement.xlsx](https://docs.google.com/spreadsheets/d/1mOW1FroilbJDNzBE389YNFgohYFefJFw/edit?usp=sharing&ouid=105474702096919495953&rtpof=true&sd=true)

## Timeline (this section should be filled in iteration 0 and updated at the end of each later iteration)

| Iteration | Functional Requirements | Tasks (Cross requirements tasks) | Estimated/real person hours |
| --- | --- | --- | --- |
| 1 | * Search for Videos (Essential) * Video watching experience (Essential) | * Design the homepage with a search bar to search for topics * Add input validation to the search bar * Add logic to the search bar to fetch relevant topics using the YouTube API * Store search history in the database * Provide a list of videos for the user to pick from * Implement a media player for user to watch videos * Provide ability for user to skip to next video or quit watching a video | 25 hrs |
| 2 | * Authentication and Profile Management (Desirable) * Analytics Dashboard (Essential) | * Update users watch time to database * Calculate users attention span * Implement Google Auth with Firebase * Create users profile page with ability for user to add his hobbies and interests * Create analytics dashboard with graphs for watch time on educational videos, entertainment videos * Create analytics dashboard with graphs for users attention span while learning and while watching entertainment videos | 40 hrs |
| 3 | * Earn points by staying focused (Desirable) * Entertainment feed to redeem user points (Desirable) * Share progress with friends (Optional) * Mobile friendly (Optional) | * Calculate points to be given to user as per users watch time * Update calculated points to firestore * Create an entertainment feed to redeem user points * Implement a share functionality to create a snapshot of dashboard to share with his friends * Make the web-app more mobile friendly | 67 hrs |

# Configuration Management Plan

## Tools

We will use Git/Github as the version control tool for our project. We plan to use JIRA for planning sprints, distributing tasks among team members and tracking and creating new issues/bugs.

We have chosen VSCode as the IDE to develop and maintain the code on our local machines. The reason for selecting VSCode is because it provides extensions for both Python and React frameworks.

We also plan to containerise our application using docker.

We will leverage the following tools provided by GitHub under the Actions tab on the repository to set up the workflow of our code: (This is subject to change as we progress with our learning as a team)

* RedHat’s Openshift tool for deployment.
* CodeQL is a SAST tool that helps discover vulnerabilities across a codebase. This tool works on codebases written in several different languages including JavaScript.
* Pysa is a tool that provides static analysis for Python code.
* APISec tool to allow continuous testing of API Security.
* SonarQube is a DAST tool that we’ll be using.
* Github-action-tester for automating the testing process.
  1. Code Commit Guideline and Git Branching Strategy[4]

We will make use of the GitHub Flow branching strategy. The reasons for choosing this branching strategy are as follows:

* GitHub Flow focuses on Agile Principles, and our team will be using the Scrum Agile framework to manage our project.
* We need quick and continuous deployment from our master branch, and this strategy allows us to do so.
* The strategy is suited for small teams such as ourselves.

For every piece of code that we commit to the repository, we will prefix the commit message with the corresponding JIRA ticket ID.

Every feature/JIRA ticket will have a separate branch. Once the changes for that branch feature/issue are complete, a pull request from that branch to the main/master branch will be created.

Ideally, we want the CI/CD checks to be carried out on each PR. Once all mandatory checks have been successful, the code can be reviewed and merged into the main branch.

For merging, we’re going to use the ‘Squash and Merge’ option provided by git. The resulting commit will be prefixed with the JIRA ticket ID, and the JIRA ticket description will be the commit message. We’re choosing this approach because we want each commit on the main branch to be executable and deployable since we will not have a separate release branch.

## Deployment Plan if applicable

Note: Our current deployment plan is subject to change as we learn and gain more experience with the tech stack and integrate them together.

We want the process of deployment to be automated. Ideally, we’d like to deploy each commit of the main branch once we have a method of communication between the backend (Flask API) and the frontend (React app).

The Deployment plan can be broken down into 4 parts[5]:

1. First, we will containerise the flask application. The Flask application will provide us with APIs that the React application leverages. We’ll build an image of the Flask app. This is the backend/API container.
2. We will build a separate image for the React application. This can be called the frontend/Client container.
3. We will upload both our container images to the openshift container registry. Create a docker-compose.yml with two services: api and client, each referring to the respective containers.
4. We set the provider for running docker-compose up as openshift and OpenShift will provide a URL through which we can access the front end of our application.

# Quality Assurance Plan

## Metrics[6]

| Metric Name | Description |
| --- | --- |
| Test metric | As part of test metric we track   1. No of test cases written 2. Ratio of passed vs failed test cases |
| Coverage metric | Requirements test coverage, a metric to measure the percentage of requirements that have been tested.  Test coverage metric tells us the percentage of code that has been tested in the entire codebase |
| Defect density metric | To monitor the number of defects we have in each module |
| Performance metric | Response time would be our primary performance metric to track the overall time taken by the application to respond to the user actions. |
| Release metric | As part of release metric we track 1. No of releases completed  2. Release deployment success rate  3. No of defects found post release |
| Cost metric | As part of cost metric, we track the estimated vs actual time required to complete task |

* 1. Coding Standard[7]

Our code will be compliant with the following coding standards:

1. Naming convention:
   1. Variables and functions: lower camel case
   2. Constants: screaming snake case
   3. Classes: upper camel case
2. Indentation: use tabs in Python, and an opening brace at the end of a line while a closing brace would be on a new line.
3. Well-documented code: Add a comment describing the purpose of a function and its arguments, in case it isn’t self-explanatory.
4. Commented headers for different files - a short description of what functionality is included in the file (optional)

## Code Review Process

As mentioned in Section 5(b), the code review process will begin after the CI/CD checks have passed.

Every pull request needs at least 2 approvals. Any 2 out of 4 of the following leaders can approve a pull request: the Requirements Leader, the Design Implementation Leader, the Security Leader and the QA Leader. When needed, the PR submitter can request more than 2 approvals before merging their changes.

For ease of the approval process, if the PR submitter had been using the pair programming technique, then their pair may submit an approval with a comment mentioning the same.

Each PR will have a comment that describes the testing done on the code changes. We may include a checklist at a later stage.

A reviewer should provide feedback on code standards, testing, and any issues they find in the code.

## Testing

Manual Testing:

Each team member is responsible for adding unit tests for the feature additions, bug fixes they work on.

We’ll use the library pytest to write our test cases and Cypress for testing our application’s UI.

The QA leader will be responsible for adding integration tests.

Automated testing:

We’ll be using github-action-tester for automatically running the test cases for each new pull request, or the case when there is a new commit to an existing pull request.

## Defect Management

We will be using JIRA for issue/defect tracking. A new ticket will be created in case any team member finds a defect in the code. The defect needs to be a significant one in order to warrant assigning it to another team member. In case of a minor defect or issue, the team member may fix the issue with their own changes.

In case the defect requires changes that are more than 1-2 lines of code, the issue will be reported to the *Issue/Defect Manager.*

An Issue Manager would change on a weekly basis. The responsibilities of an Issue Manager would be:

* Assigning new features/issues to a team member
* Taking meeting minutes for that week
* Organising scrum meetings (on discord chat) on a regular basis (once in 2-3 days)
* Ensuring PRs are getting reviewed on time.

# References

[1] Jennifer Oaten. “Decreasing Attention Spans And The Effects On Learning”, *LinkedIn,* August 26, 2022, <https://www.linkedin.com/pulse/decreasing-attention-spans-affects-learning-jennifer-oaten>

[2] Trishala Chokhani. “A Deep Dive Into The Shrinking Attention Span”, *E-learning Industry,* February 23, 2023, <https://elearningindustry.com/a-deep-dive-into-the-shrinking-attention-span>

[3] Nicole Gallucci. “Forest is a useful app that helps you go phone-free by inspiring you to plant trees”, *mashable,* May 2, 2019, <https://mashable.com/article/forest-app-productivity-focus-review>

[4] Haddad, Rowan. “What Are the Best Git Branching Strategies”, *Abtasty,* March 8, 2022, [www.abtasty.com/blog/git-branching-strategies/](http://www.abtasty.com/blog/git-branching-strategies/)

[5] Grinberg, Miguel. “How to Dockerize a React + Flask Project” *Blog,* June 13, 2021, <https://blog.miguelgrinberg.com/post/how-to-dockerize-a-react-flask-project>

[6] “A Detailed Guide on Agile Software Testing Metrics in 2023”, *testingxperts*, Jan 20, 2023, <https://www.testingxperts.com/blog/software-testing-metrics-guide>

[7] Kumar, Sayan. “Coding Standards and Guidelines”, *geeksforgeeks*, June 28, 2022, <https://www.geeksforgeeks.org/coding-standards-and-guidelines/>

# Glossary

(Any acronym used in the document should be explained here)