FEELTOK

SWE6202 – AGILE PROGRAMMING

REPORT DONE BY:
LES PAUL RANALAN
(2230727)

TO BE SUBMITTED TO:
RENUKA NYAYADHISH

DATE OF SUBMISSION:
DECEMBER 19, 2024

TABLE OF CONTENTS

TABLE OF FIGURES

INTRODUCTION

This report documents my experiences in an Agile software development project in a group setting, focusing on the creation of software artifacts and their documentation. Using Agile practices, we developed a social media platform where people can share their feelings with others, ensuring collaboration and incremental progress. My role includes front-end development, back-end development, and sprint planning.

The report explores the practices that we used, the team dynamics, the challenges that we faced, and the lessons we learned throughout the project.

WHY AGILE APPROACH

The agile approach is a flexible, iterative methodology that is mostly used in project management and software development **(Laoyan, 2024)**. It delivers value more efficiently as it emphasizes adaptability, collaboration, and continuous progress. This allows teams to respond to change quickly while focusing on customer needs.

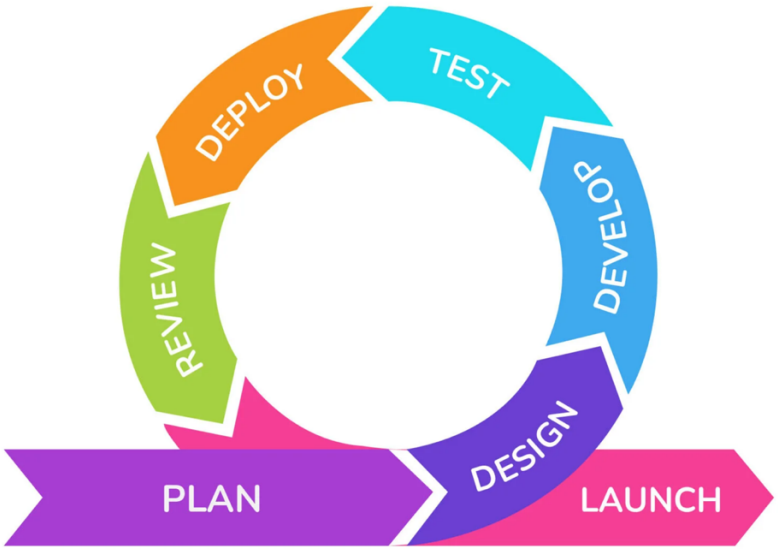


Figure 1: Agile methodology model

When developing FeelTok, the agile approach was selected because it allowed the team to stay adaptable and responsive where user requirements and market demands could change over time. By working in iterative sprints, there were increments in functionalities being developed. Over time, feedback was gathered from stakeholders and continuously refined the final app. Agile's emphasis on collaboration ensured that the team worked together cohesively as daily meetings were conducted. Additionally, retrospectives helped the team improve in dealing with specific challenges. This approach enabled the team to create FeelTok, with all of its user requirements implemented.

PROJECT VISION

The vision of the project was to create a new mobile application that is designed to help people reconnect and nurture real human relationships, both personal and professional. It revolves around 6 emotional pillars which are **Love**, **Gratitude**, **Apology**, **Appreciation**, **Mindfulness**, and **Thankfulness**. The app will offer a simple, heartfelt space where users can post texts and images. Through this, users who are involved in FeelTok will have greater emotional well-being and FeelTok will build on that trust as the company that promotes positive social impact.

HARDWARE & SOFTWARE REQUIREMENTS

Hardware and software requirements define the technical specifications needed to execute and complete the app **(Knotts, 2024)**. The requirements ensure that development goes as smoothly as possible and that the final app is fully functional and meets all the requirements.

Below are the hardware and software requirements of FeelTok:

HARDWARE:

- Computers/laptops with 8GB and i5 processors or greater
- Android device/emulator
- iOS device/emulator
- Cloud-based server

SOFTWARE:

- VSCODE
- React Native
- Expo
- Firebase
- EmailJS
- Jira
- Zephyr Scale

With the mentioned requirements above, both the development and testing of FeelTok can be executed smoothly.

RISK MANAGEMENT STRATEGIES

Risk management strategies are approaches used to identify and mitigate potential risks that could affect the completion of the app. These strategies aim to minimize the uncertainties of deadlines, quality, and project results **(Gibson, 2023)**. This promotes the team to be adaptable to changes.

ID	Current Status	Risk Impact	Probability of Occurrence	Risk Map	Risk Description	Project Impact	Risk Area	Symptoms	Triggers	Risk Response Strategy	Response Strategy	Contingency Plan
R001	Open	High	Medium	Yellow	Data Security Issues	Loss of user data, damage to reputation, legal problems.	Technical	Users notice strange data or unauthorized access.	Unusual activity on user accounts.	Prevent & Avoid	Improve app security, use encryption and secure logins.	Add multi-factor authentication and perform security checks regularly.
R002	Open	Medium	High	Yellow	Low User Interest	Slower growth, harder to get enough users, and low app engagement.	Market	Few people signing up or using the app, negative feedback.	Drop in sign-ups and app usage.	Avoid	Get user feedback, test app improvements.	Run a marketing campaign, offer rewards for new users.
R003	Open	Low	Medium	Green	Server Problems	Temporary app problems, but won't cost much.	Operational	App slows down, shows error messages during busy times.	Issues during busy hours.	Prevent & Avoid & Transfer	Upgrade servers and balance loads better.	Set up backup servers to prevent problems.
R004	Open	High	High	Red	Privacy Issues	Users lose trust, could lead to fines or loss of users.	Legal & Compliance	Unclear privacy policies, user complaints about data sharing.	Change in data rules or privacy concerns.	Prevent	Review privacy policies and make sure they're clear.	Quickly update privacy policies and inform users.
R005	Open	Medium	Medium	Yellow	App Problems	Service interruptions and delays in new updates.	Technical	Bugs, crashes, or features not working.	More support requests and complaints.	Prevent	Improve testing, automate bug fixes.	Release updates and bug fixes regularly.
R006	Open	Low	Low	Green	Lack of User Engagement	main features are underused, lower app activity.	Operational	People don't interact much with emotion-sharing features.	Declining activity from users.	Accept & Prevent	Improve the user experience and provide better reminders.	Send users reminders and notifications to share emotions.
R007	Open	High	Low	Yellow	Legal Compliance Issues	Risk of lawsuits or fines if the app breaks any laws.	Legal & Compliance	Not keeping up with updated laws or regulations.	Missed deadlines for legal updates.	Avoid	Keep the app up-to-date with legal standards.	Review app's terms of service and privacy policies.
R008	Open	Medium	Medium	Yellow	Budget Overrun	Delays or resource problems if costs go over budget.	Organizational	Costs exceeding expectations, running out of resources.	Higher costs than planned.	Avoid & Prevent	Keep an eye on spending, adjust resources as needed.	Cut down on non-essential tasks, delay some features.
R009	Open	Low	High	Yellow	New Competitors	Losing users if other similar apps launch.	Market	Users moving to better apps or apps with more features.	More people downloading competitors' apps.	Prevent	Focus on unique features and improve marketing.	Add new features, adjust pricing or run ads.
R010	Open	Medium	Medium	Yellow	Employee Turnover	Delays if important team members leave.	Organizational	Drop in productivity or unhappy employees.	Higher turnover or difficulty filling open positions.	Prevent	Boost team morale and engagement.	Hire temporary workers or contract developers.
R011	Open	High	Medium	Yellow	Third-Party Service Failures	Problems with external services like APIs or payment systems.	Technical	Services stop working or experience errors.	Third-party service issues affecting the app.	Transfer	Make sure contracts with third-party services are clear.	Have backup services ready in case of failure.
R012	Open	Low	Low	Green	Natural Disasters or External Crises	Natural disasters or global issues delaying the project.	External	Project delays due to outside events.	A major event like a natural disaster or global crisis.	Accept	Keep track of outside factors, communicate with users about delays.	Set up backup plans to work remotely or recover from events.
R013	Open	Medium	Medium	Yellow	Low Team Motivation	Team becomes less productive, project gets delayed.	Organizational	Decreased energy, lack of initiative, unhappy team members.	Drop in performance or missed deadlines.	Prevent	Hold regular check-ins and recognize good work.	Set clear goals, offer rewards, or change roles if needed.
R014	Open	High	Medium	Yellow	User Addiction	Users spend too much time on the app, affecting their well-being.	Market	Users feel overwhelmed or are too focused on the app.	Users report feeling addicted or spending too much time on the app.	Prevent	Add features to encourage healthy use, like usage limits.	Introduce time limits, promote well-being, and educate users on healthy app use.

Figure 2: Risk management log

Below are the identified risks and their mitigation of FeelTok:

IDENTIFIED RISK	**MITIGATION**
Data security issues	Implement secure logins and encryption
Low user interest	Get user feedback and test app improvements
Third-party service failure	Have a backup of alternative services
Privacy issues	Review privacy policies and let users know what data is being used
App bugs	Improve testing and fix bugs
Lack of user engagement	Improve user experience and provide reminders

Legal compliance issue	Keep the app up-to-date with legal standards
Budget overrun	Keep an eye on spending, and adjust resources if required.
New competitors	Focus on unique features and improve marketing
Team turnover	Boost team morale and engagement
Low team motivation	Hold regular meetings and recognize good work
User addiction	Introduce time limits and remind users how many hours they have used the app

When developing FeelTok, the following risks were faced and mitigated:

- **App bugs** - the team mitigated this risk by testing, debugging the app, and eventually fixing the bugs or unintended behavior.
- **User addiction** - the team mitigated with risk by deciding to implement a timer system to avoid user addiction when using the app.
- **Low team motivation** - the team mitigated this risk by initiating a meeting with each team member, not only by the scrum master, and providing sources such as documentation and video tutorials that the team can follow. Additionally, the team recognized good work on team member's outputs.

PRODUCT BACKLOG

Product backlog is a list of tasks or features that are required to develop and improve the app. It serves as a dynamic to-do list that evolves throughout the entire development duration **(Scrum.org, 2019)**.

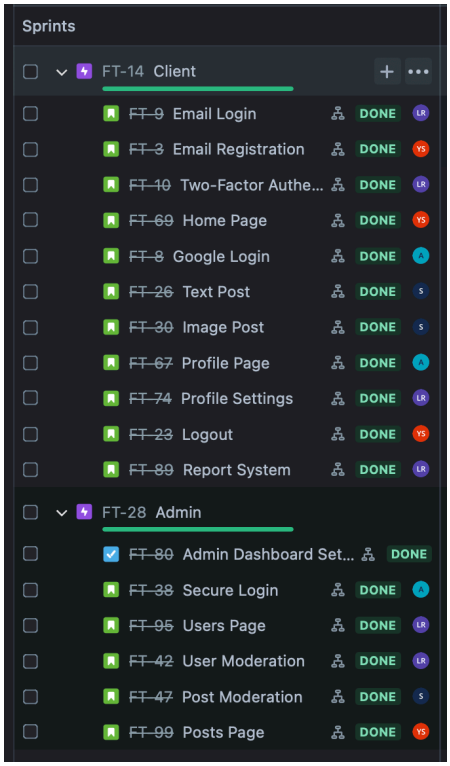


Figure 3: Product backlog in Jira

Below are the product backlogs of FeelTok's client and admin:

CLIENT:

FEATURE	**DESCRIPTION**
Email Login	The user should be able to sign in via email
Email Registration	The user should be able to sign up via email

Two-Factor Authentication	The user should have the option to have an extra layer of security
Home Page	The user should be able to view other people's posts
Google Sign-in	The user should be able to easily sign in via a Google account
Text Post	The user should be able to create a post that consists of texts
Image Post	The user should be able to create a post that consists of both text and an image
Profile Page	The user should be able to view their current profile details
Profile Settings	The user should be able to edit or modify their profile or account
Sign-out	The user should be able to exit the current session of the app
Report System	The user should be able to report both users and posts
Timer System	The user should be able to be reminded and monitor how many minutes or hours the user has been using the app

ADMIN:

FEATURE	DESCRIPTION
Secure Login	The admin should be able to securely sign in to the dashboard
Users Page	The admin should be able to view all users and their details
User Moderation	The admin should be able to ban users by viewing user reports
Posts Page	The admin should be able to view all posts and their details
Post Moderation	The admin should be able to delete posts by viewing post reports

With the entire backlog mentioned above, we can view the entire features list with its description. Through this, we can then group features to create a release plan for quick deployment, identify which features can be prioritized or MVPs, and eventually plan which features to implement for each sprint.

RELEASE PLAN

A release plan is a roadmap that outlines the timeline, scope, and deliverables of an upcoming release **(monday.com, 2022)**. It connects the gap between the vision and the work done in individual sprints. This helps the team in aligning with the project goal.

Below are the release plans for FeelTok:

- **VERSION 1.0** - Basic screens and navigation, with Firebase password authentication
- **VERSION 1.1** - Profile Editing
- **VERSION 1.2** - Posts and Google sign-in
- **VERSION 1.3** - Bug fixes and timer system
- **VERSION 1.4** - Finalize app

With the mentioned features in each version above, we can see which features are to be implemented for each version per release. The team was able to execute the release plan with version 1 being the most basic, and each version having progress increments, until the very last version with bug fixes.

SPRINT PLANNING

Sprint planning is a meeting that is conducted at the start of a sprint when in an Agile project **(Scrum.org, 2000)**. The meeting defines the goals, tasks, and scope of the work to be completed within the sprint period. This ensures that team members are aware and aligned on the tasks to be done and the sprint's timeframe.

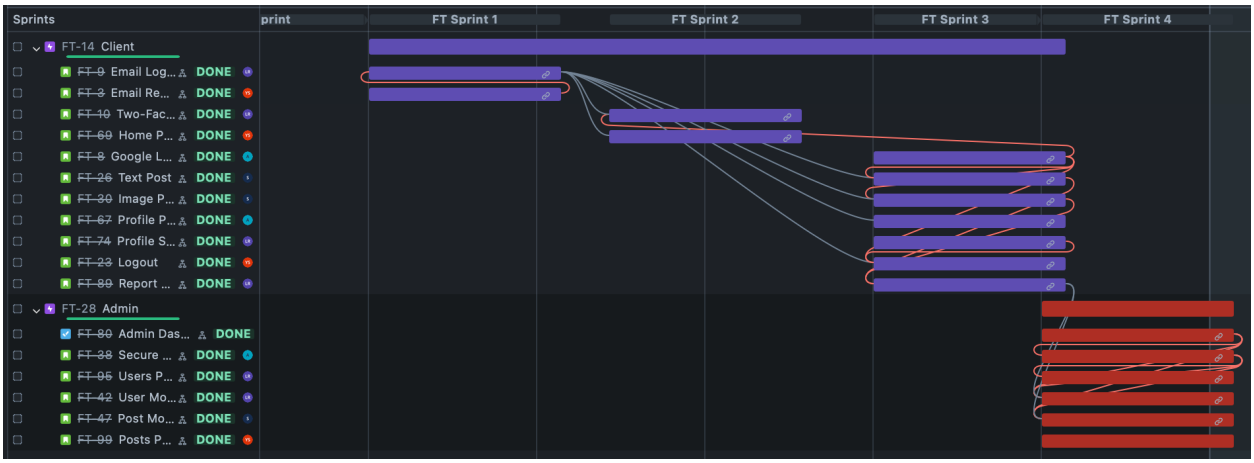


Figure 4: Sprint planning in Jira

Below are the sprint planning of FeelTok:

- **SPRINT 1:** Client app setup, Firebase database setup, email authentication
- **SPRINT 2:** Two-factor authentication, home page
- **SPRINT 3:** Google sign-in, text posts, image posts, profile page, profile settings, sign-out
- **SPRINT 4:** Admin app setup, users page, user moderation, posts page, post moderation, timer system

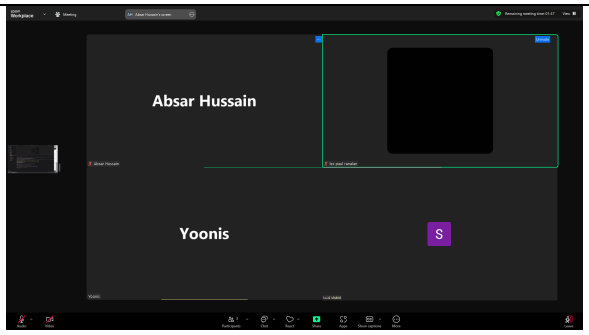
With the screenshot above, some features were moved onto the next sprint due to incompletion or technical errors. All features of planned sprints were eventually completed at the very last sprint.

DAILY STAND-UP MEETING

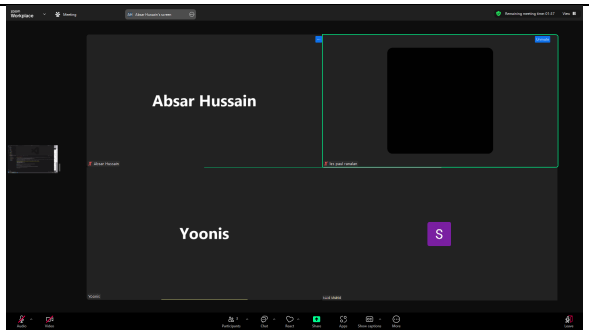
The daily stand-up meeting is a meeting conducted every day for the entire sprint duration. Each member actively updates the team on what they are working on and what is considered done **(Geekbot, 2020)**.

When developing FeelTok, stand-up meetings were conducted for each team member to be updated on what is currently in progress, and which functionalities are done. Below are the evidences of stand-up meetings of FeelTok:

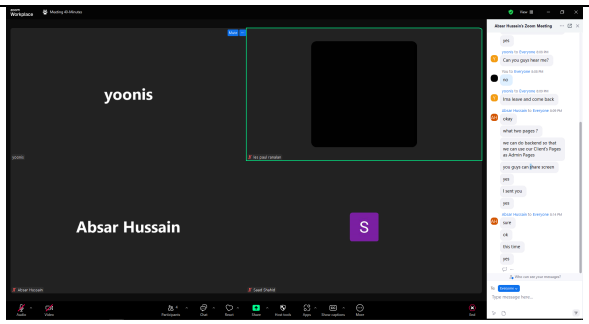
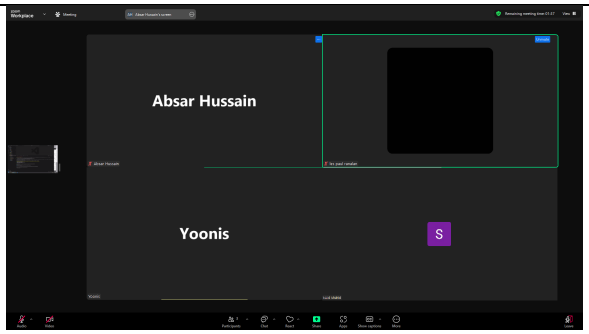
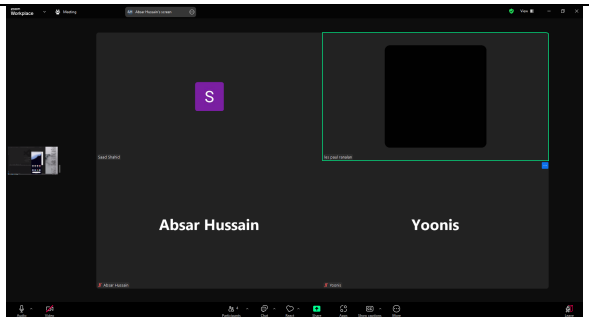
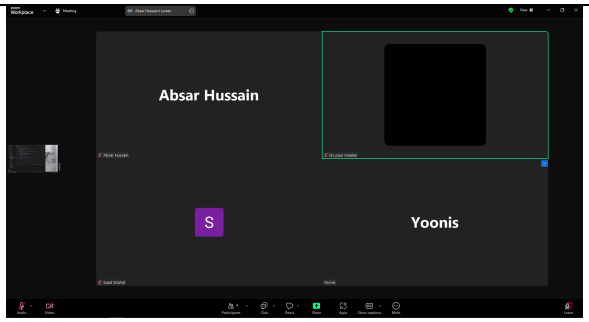
SPRINT 1:

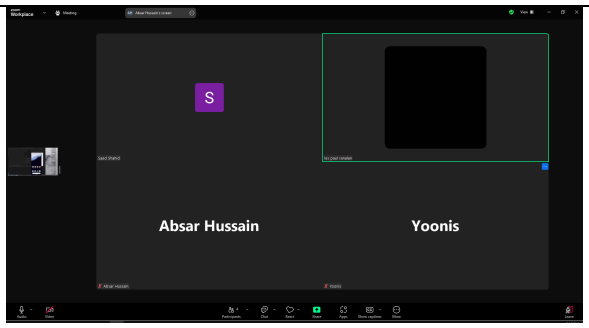
MEETING NUMBER	**EVIDENCE**
Meeting 1	N/A
Meeting 2	N/A
Meeting 3	N/A
Meeting 4	Conducted in-class
Meeting 5	Conducted in-class
Meeting 6	
Meeting 7	N/A

SPRINT 2:

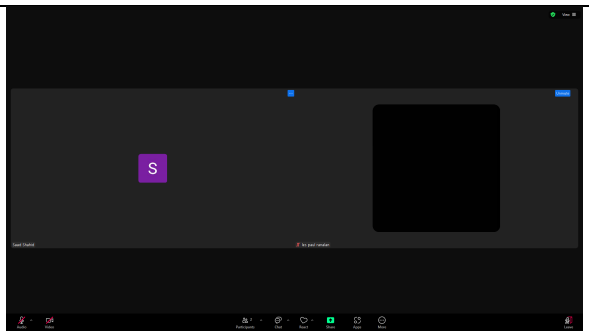
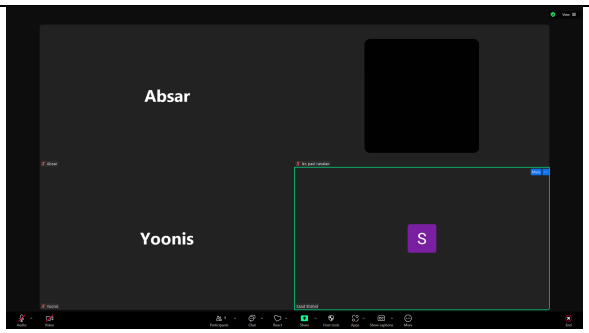
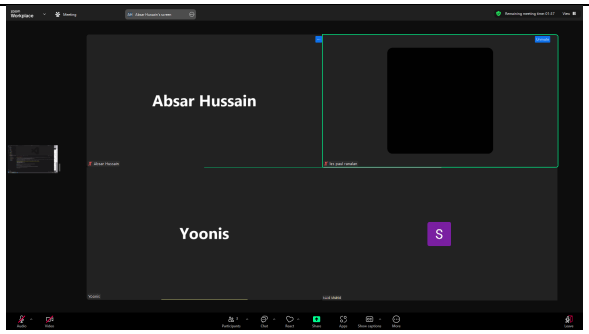
MEETING NUMBER	EVIDENCE
Meeting 1	N/A
Meeting 2	
Meeting 3	N/A
Meeting 4	N/A
Meeting 5	Conducted in-class
Meeting 6	Conducted in-class
Meeting 7	N/A

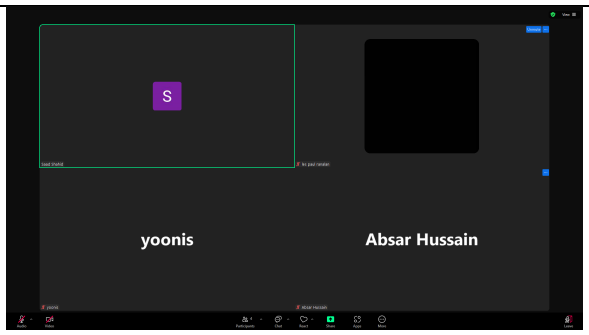
SPRINT 3:

MEETING NUMBER	EVIDENCE
Meeting 1	
Meeting 2	
Meeting 3	
Meeting 4	
Meeting 5	Conducted in-class
Meeting 6	N/A

Meeting 7	

SPRINT 4:

MEETING NUMBER	EVIDENCE
Meeting 1	
Meeting 2	N/A
Meeting 3	
Meeting 4	Conducted in-class
Meeting 5	Conducted in-class
Meeting 6	

Meeting 7	

Each sprint was executed for one week, hence the seven meetings. Daily stand-up meetings were executed on some days, but none on other days due to external reasons (holidays, absence, lack of communication, etc).

PRIORITY FEATURES & TASKS

Priority features and tasks refer to the most important and urgent functionalities in an Agile project that need to be completed as a priority. It is very prevalent in Scrum or Kanban frameworks, which are typically defined and managed through backlog, sprint planning, and user story prioritization.

When developing FeelTok, four major features were considered MVP (minimum viable product) or top priority:

- **User Authentication**, which lets the user sign in and sign up for the app itself.
- **A basic user interface**, which lets users see all posts from other users.
- **Posting**, which lets the user create and share text-based or image-based posts with others.
- **Moderation**, which lets the admin monitor and exercise necessary actions on the app.

The mentioned MVPs above are basic functionalities that are NEEDED for FeelTok to function properly. The advanced features can then be added later on within sprints, after feedback from stakeholders or users.

USER STORIES

User stories are concise descriptions of a feature that a user needs in a software app **(Visual Paradigm, 2019)**. They are an important element when it comes to agile projects, as they help teams focus on delivering value from the perspective of the user.

Below are the client and admin user stories of FeelTok:

CLIENT:

USER STORY	USER ACCEPTANCE CRITERIA
Email Login: *"As a user, I want to be able to log in using my e-mail so that I can use FeelTok."*	• The user must enter a valid email with a matching "@" symbol in the email input field. • The user must enter a valid password with minimum characters in the password input. • The user should have a button to verify user credentials. • The user should be redirected after successful sign-in. • The user should be alerted if sign-in fails.

Email Registration: *"As a user, I want to register using my e-mail so that I can use FeelTok."*	• The user must enter a valid email address in the email input field, including the "@" symbol. • The user must create a password with a minimum of 8 characters in the password input field. • The user should be able to click the "Register" button to submit the registration form. • The user should receive a confirmation email after successful registration. • The user should be redirected to a confirmation page or login screen after successful registration. • The user should see an error message if the email is already registered or if the registration fails.
Two-Factor Authentication: *"As a user, I want to be able to have two-factor authentication on login so that my account will be more secure."*	• The user should be on the OTP screen if 2FA is enabled on the user's account. • The user should be able to enter numerical characters only. • The user should have a button to confirm OTP. • The user should be redirected to the Home page if 2FA is successful. • The user should be alerted if 2FA fails.

Home Page: *"As a user, I want to see other user's posts so that I am up-to-date on their activities."*	• The home page must load successfully without errors upon user login. • The home page should display a navigation bar with links to essential sections like Profile, Settings, and Posts. • The user posts should be visible in a well-organized feed, with each post displaying the user's name, profile picture, and post content.
Google Sign-in: *"As a user, I want to be able to log in using my already existing Google account so that I can easily sign in to FeelTok."*	• The user should have a button to sign in using a Google account. • The user should be able to cancel or continue with Google when the pop-up is shown. • The user should be redirected to the Home page if Google Sign-in is successful. • The user should be alerted if Google Sign-in fails.
Text Post: *"As a user, I want to post texts on FeelTok so that I can share my thoughts with other users."*	• The user should enter valid caption text, not leaving the caption input field blank. • The user should have a button to create a post. • The user should be alerted if a post is created successfully. • The user should be alerted if creating a post fails.

Image Post: "*As a user, I want to upload and share images so that I can visually share my experience with other users."*	• The user should have a button to upload an image. • The user should have a button to create a post. • The user should be alerted if creating a post is successful. • The user should be alerted if creating a post fails. • The user should be able to post without caption, but with image only.
Profile Page: *"As a user, I want to view my profile so that I can see my posts."*	• The user should see all of their posts. • The user should have a button to go to Profile Settings. • The user should see basic information such as username, name, profile picture, bio, etc.

Profile Settings: *"As a user, I want to change my user credentials so that I can customize my profile."*	• The user should see all of the user's details such as name, email, profile picture, etc. • The user should have a button to edit their details. • The user should have a button to delete their account. • The user should have a toggle to enable or disable 2FA. • The user should enter their password for verification when changing details or deleting an account. • The user should enter valid values in the input fields. If email, match with the "@" symbol. If the password matches with minimum characters. • The user should have a button to save changes to the profile. • The user should have a button to import an image to the profile picture. • The user should have a button to remove the profile picture. • The user should have a dropdown for gender. • The user should be alerted if any operation (editing, deleting) is successful or failed.

Sign-out: *"As a user, I want to log out and exit the application so that my credentials are not saved and are secured."*	• The logout button must be visible in the navigation or profile menu. • Upon clicking the logout button, the user must be logged out of the application. • After logging out, the user should be redirected to the login page. • If the user attempts to access a protected page after logging out, they should be redirected to the login page.
Report System: *"As a user, I want to report inappropriate posts or users so that FeelTok can maintain its positive integrity."*	• The user should see a report button when clicking the three-dots option in each post. • The user should enter valid text in the input field. • The user should have a button to send the report. • The user should be alerted if send report is successful • The user should be alerted if sending a report fails.
Timer System: *"As a user, I want to be reminded how many hours or minutes I have been using the app so I can take a break from FeelTok.*	• The user should see a timestamp in the top-right of the home page, showing how many hours or minutes they have been using the app. • The user should be alerted if the user has been using the app for too long. • The user should have the option to ignore if the alert is present.

ADMIN:

FEATURE	USER STORY	USER ACCEPTANCE CRITERIA
Secure Sign-in	*"As an admin, I want to log in to the admin dashboard so that I can view FeelTok's data."*	• The admin should enter valid credentials, with no input fields being blank. • The admin should have a button to sign in. • The admin should be redirected to the Home page of the dashboard if sign-in is successful. • The admin should be alerted if sign-in fails.
Users Page	*"As an admin, I want to see a list of all users and their information so that I can monitor and manage accounts easily."*	• The admin should see all of the users in table form, along with their user details. • The admin should see users who are banned in red text.
User Moderation	*"As an admin, I want to ban/restrict certain users based on reports so that FeelTok can maintain its positive integrity."*	• The admin should have a button to ban specific users. • The admin should see user reports in table form. • The admin should have a button to resolve a user report.

Post Moderation	*"As an admin, I want to review posts that were reported by users so that I can monitor which posts are following post guidelines."*	• The admin should see all of the post reports in table form. • The admin should have a button to delete specific posts. • The admin should have a button to resolve post reports.
Posts Page	*"As an admin, I want to see a list of all posts and their information so that I can have a comprehensive idea of what users are posting."*	• The admin should see all of the posts in table form, along with each post's details. • The admin should have a button to view the post in the eyes of the user.

The above user stories were used to implement each functionality, in the eyes of the user. This helps make the overall system more intuitive and promotes design to being user-centric design.

PROGRAMMING DESIGN STRATEGIES

Programming design strategies are approaches and methodologies used to organize the development process of a software app **(GeeksforGeeks, 2024)**. It ensures that the app is efficient, maintainable, scalable, and able to meet the requirements. It also guides developers in deciding how data flows, how components interact, and how the overall system is built.

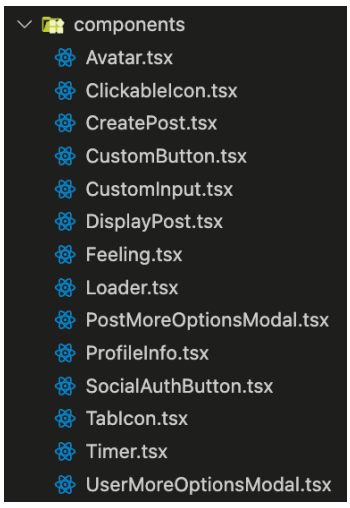


Figure 5: TypeScript components for repeated elements

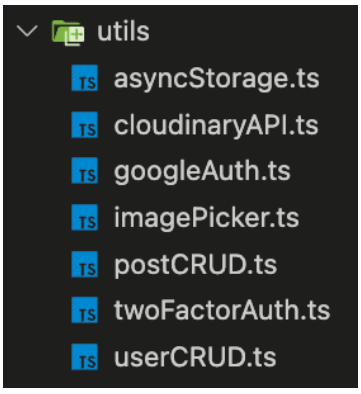


Figure 6: TypeScript files for functions and methods

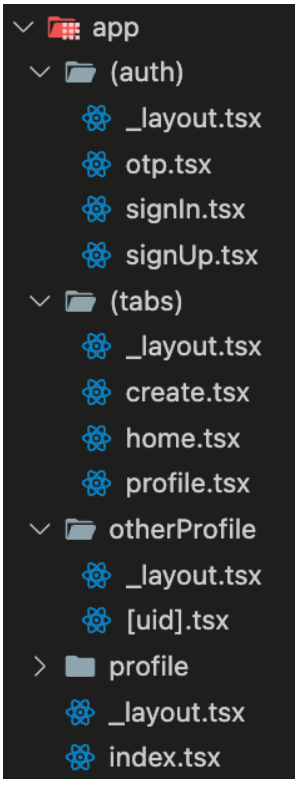


Figure 7: TypeScript components for pages and screens

To develop FeelTok, we adopted a component-based and Object-Oriented design approach to ensure the modularity and scalability of the project since each feature was encapsulated within a reusable component and uses repeated operations such as CRUD.

Within that, cloud serverless architecture was used for the project as data should be accessible everywhere with an internet connection. A user-centric design approach was also adopted to guide the user on how the app works by promoting intuitiveness and simplicity. Additionally, we adopted cross-platform development with the philosophy, "learn once, write anywhere", which allows us to create one codebase and build two apps together for Android and iOS.

PROGRAMMING CONSTRUCTS IMPLEMENTATION

Programming constructs implementation refers to the code application of programming concepts such as conditionals, try-catch, functions, etc. within an app to achieve functionality. It often involves translating design and user requirements into code via frameworks and preferred programming languages.

Using the design strategies mentioned above, FeelTok utilizes five different frameworks and services. Below are the five mentioned:

(next page)

- **FeelTok uses React Native** to properly implement each component using useStates and useEffects to handle different values and lifecycles. An example of this would be the Sign-in Screen wherein user inputs are stored within states. Additionally, React Native Navigation for navigating within the app through different pages and screens.

```
export default function SignIn() {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");

  const [isDisabled, SetIsDisabled] = useState(true);
  const [isLoading, setIsLoading] = useState(false);

  async function handleSignIn() {
    setIsLoading(true);

    // execute sign-in verification via email and password
    const verifyUserResult = await verifyUser({
      email,
      password,
    });

    if (verifyUserResult) {
      // if sign-in is successful, redirect to home
      router.replace("/home");
    }

    setIsLoading(false);
  }
```

Figure 8: Code snippet of Sign-in Screen

- **FeelTok uses Expo** to properly access device-specific features such as image gallery, with minimal code and without the use of native code (Swift, Kotlin). Additionally, it provides a way to execute a development build in the cloud via EAS. An example of this would be in the Sign-up screen wherein it utilizes the image gallery to import an image onto the user's profile picture.

```
export async function runImagePicker(
  aspectRatioWidth: number,
  aspectRatioHeight: number
) {
  try {
    // execute ask gallery permission to user
    const galleryPermissionResult = await askMediaPermission();

    if (galleryPermissionResult === "ok") {
      // if permission is granted, execute image picker
      const popup = await ImagePicker.launchImageLibraryAsync({
        mediaTypes: ["images"],
        allowsEditing: true,
        aspect: [aspectRatioWidth, aspectRatioHeight],
        quality: 1,
      });

      if (!popup.canceled) {
        // if image is selected, return uri

        console.log(runImagePicker.name, "|", "image selected");
        return popup.assets[0].uri;
      }
    }
  } catch (error) {
    console.error(runImagePicker.name, "|", error);
    Alert.alert("Error", "Something went wrong. Please try again.\n\n" + error);
  }
}
```

Figure 9: Code snippet of runImagePicker function using Expo image picker

- **FeelTok uses Firebase**, via the React Native SDK, to handle user authentication, allowing users to sign in and sign up within the app. Firestore was utilized to store post information and other user information such as the creation date, bio, one-time passwords, profile picture reference, etc. Additionally, async programming and try-catch methods were used for each Firebase operation to handle errors effectively.

```
export async function verifyUser({ email, password }: verifyUserProps) {
  try {
    // execute sign-in with entered email and password
    const result = await auth().signInWithEmailAndPassword(email, password);

    if (result.user) {
      // if user is verified, store credentials in async storage
      await setCredentials({ email, password });

      console.log(verifyUser.name, "|", "password user verified successfully");
      return result.user;
    }
  } catch (error) {
    console.error(verifyUser.name, "|", error);
    Alert.alert("Oops!", "Something went wrong. Please try again.\n\n" + error);
  }
}
```

Figure 10: Code snippet of verifyUser function using Firebase API

- **FeelTok uses Cloudinary** for storing images such as profile pictures and post images. Though Firebase has a Storage option but is now a paid option. Hence, Cloudinary was chosen as an alternative for the project to be more budget-efficient. The image link is then referenced within Firestore. Both image upload and delete operations are implemented within Firebase methods.

```
export async function uploadImage({
  name,
  uri,
  publicID,
}: UploadProfilePicture) {
  try {
    const imageData = new FormData();

    // @ts-ignore
    imageData.append("file", {
      uri: uri,
      type: `image/${uri.split(".")[1]}`,
      name: name,
    });
    imageData.append("public_id", publicID);
    imageData.append("api_key", CLOUDINARY_API_KEY);
    // create new form data wherein we append image details and signature

    // fetch upload response using api
    const uploadResponse = await fetch(
      `https://api.cloudinary.com/v1_1/${CLOUDINARY_CLOUD_NAME}/image/upload`,
      {
        method: "POST",
        body: imageData,
      }
    );

    // jsonify the response
    const data = await uploadResponse.json();

    if (uploadResponse.ok) {
      // if response is successful, return the image url

      console.log(uploadImage.name, "|", "image uploaded successfully");
      return data.secure_url;
    } else {
      console.error(uploadImage.name, "|", "image upload failed");
    }
  } catch (error) {
    console.error(uploadImage.name, "|", error);
  }
}
```

Figure 11: Code snippet of uploadImage function using Cloudinary API

- **FeelTok uses EmailJS** to implement the two-factor authentication by sending the one-time password, generated within the app and assigned to Firestore, to the user's email inbox. Though Firebase has a phone number-based two-factor authentication option it is a piad option. Hence, an email-based two-factor authentication was used as an alternative for the project to be more budget-efficient. EmailJS is implemented within user authentication with a conditional wherein if the user has two-factor authentication enabled, it triggers the API.

```
export async function sendOtp(firebaseUser: FirebaseAuthTypes.User) {
  try {
    // assign general references
    const userDoc = firestore().collection("users").doc(firebaseUser.uid);
    const docSnap = await userDoc.get();
    let otp = "";

    if (docSnap.exists) {
      // if user exists, generate otp
      otp = Math.floor(100000 + Math.random() * 900000).toString();
    }

    // update otp value in firestore
    await userDoc.update({
      otp: otp,
    });

    // send otp to user's email via emailJS api
    await emailjs.send(
      EMAILJS_SERVICE_ID,
      EMAILJS_TEMPLATE_ID,
      {
        otp: otp,
        to_name: firebaseUser.displayName,
        to_email: firebaseUser.email,
      },
      { publicKey: EMAILJS_PUBLIC_KEY }
    );

    console.log(sendOtp.name, "|", "OTP sent to user's email");
    return "ok";
  } catch (error) {
    console.error(sendOtp.name, "|", error);
    Alert.alert("Oops!", "Something went wrong. Please try again.\n\n" + error);
  }
}
```

Figure 12: Code snippet of sendOTP function using EmailJS API

SPRINT REVIEWS & BURNDOWN CHARTS

Sprint reviews are meetings held at the end of a sprint when in an Agile project. It is to inspect the work completed during the current sprint, demonstrate the functionality of the progress increment to stakeholders, and gather feedback for improvements **(Scrum.org, 2024)**.

The team executed sprint reviews with the means to update the stakeholders on the progress of the app and to gain feedback. However, in this case, each member can take turns to become the stakeholder and have feedback based on their opinions. Through this, the app became more refined and is agreeable to everybody. Additionally, we discussed the burndown charts of each sprint, which are the work left to do versus the time it takes to complete it **(Asana, 2024)**. Below are the sprint burndown charts for each of the four sprints and the entire duration of development:

(next page)

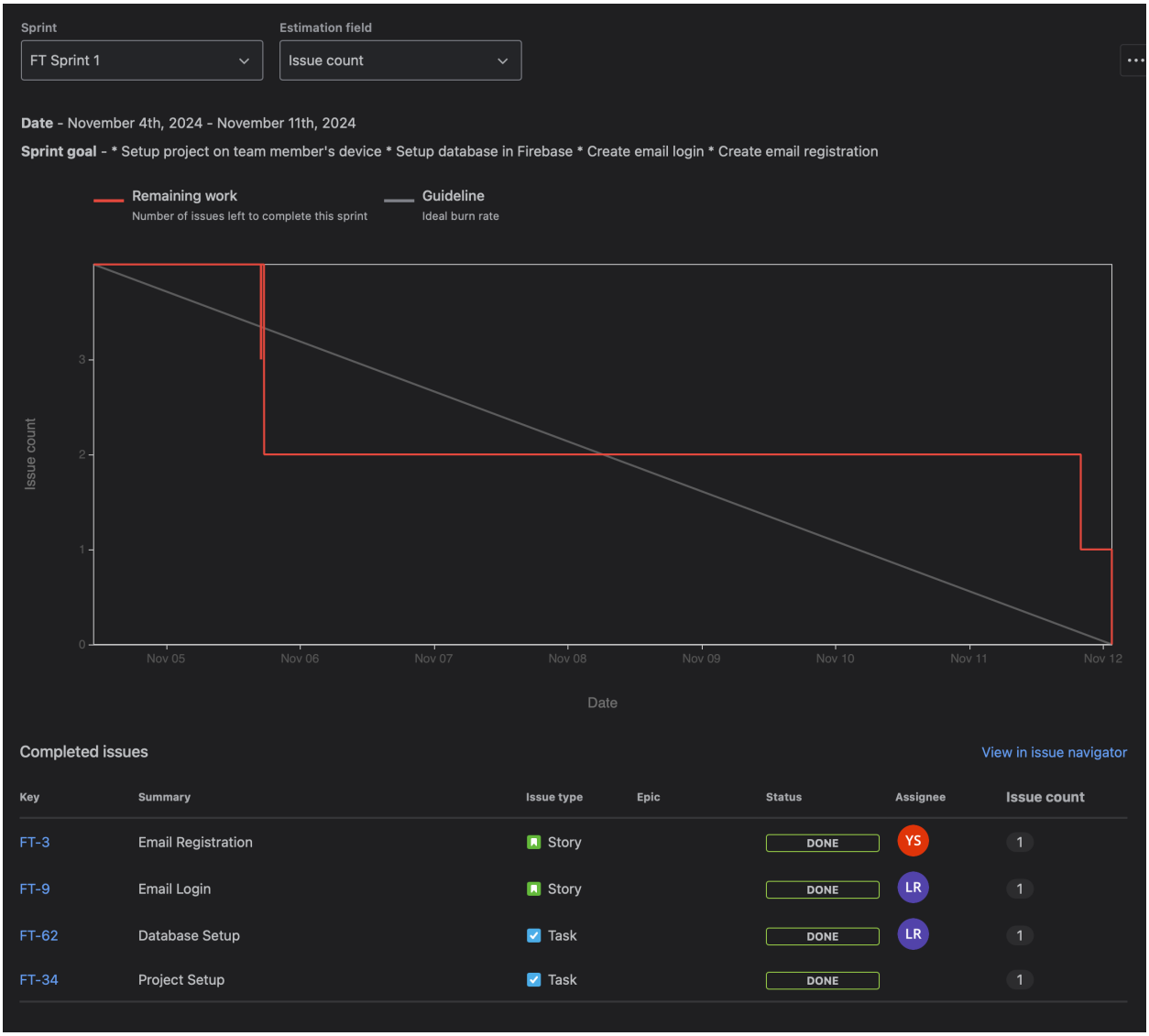


Figure 13: Sprint 1 report

As seen above, the first sprint covers Email Registration, Email Login, Database Setup, and Project Setup. All of these were completed on time and within the sprint duration.

Figure 14: Sprint 3 report

As seen above, the second sprint covers Two-Factor Authentication, Home Page, and supposedly Google Login. Most of the issues were completed, however, Google Login was incomplete till the end of the sprint, therefore it was moved towards the next sprint. There is also a case wherein the sprint was extended by one day.

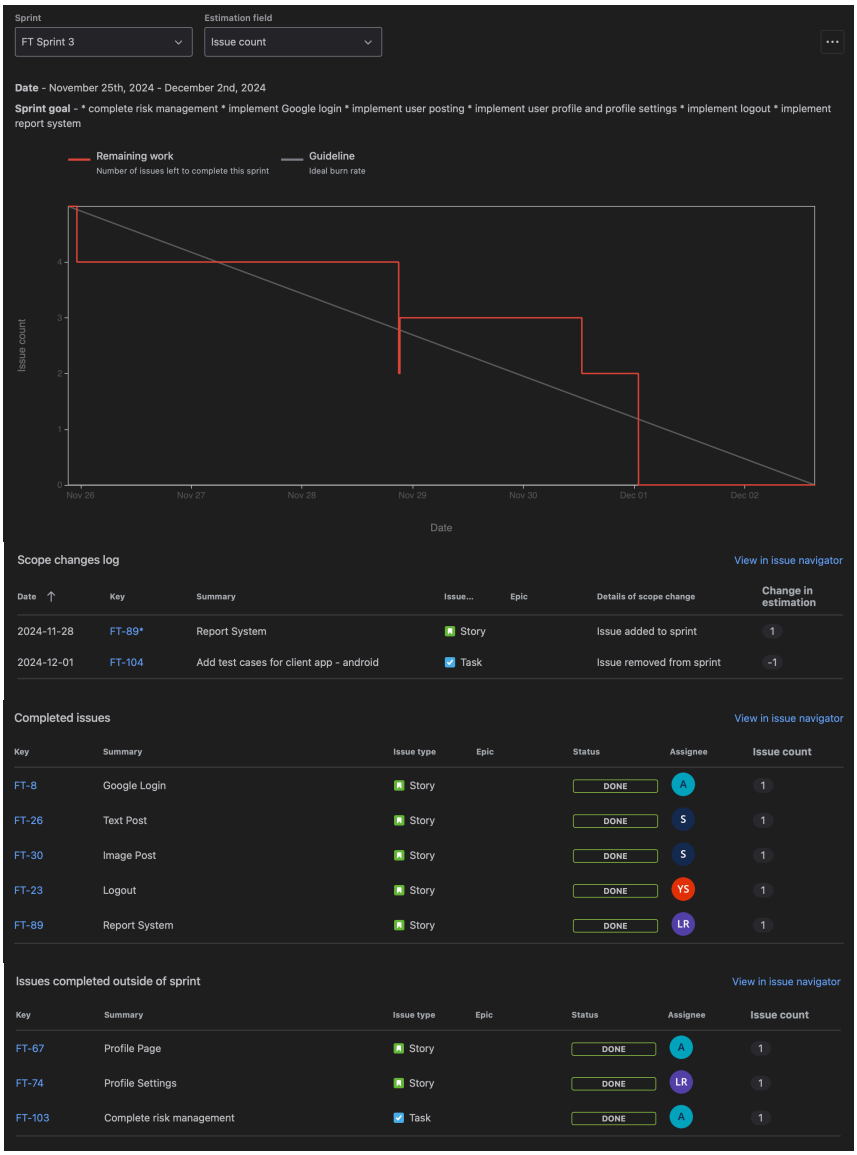


Figure 15: Sprint 3 report

As seen above, the third sprint covers Google Login, Text Post, Image Post, Log Out, Report System, and supposedly test cases for Android. All of which were completed early within the sprint. The remainder of the sprint completes three issues which are the Profile Page, Profile Settings, and completing the risk management log. However, test cases for Android were removed and moved on to the next sprint. With that, the sprint ended on schedule.

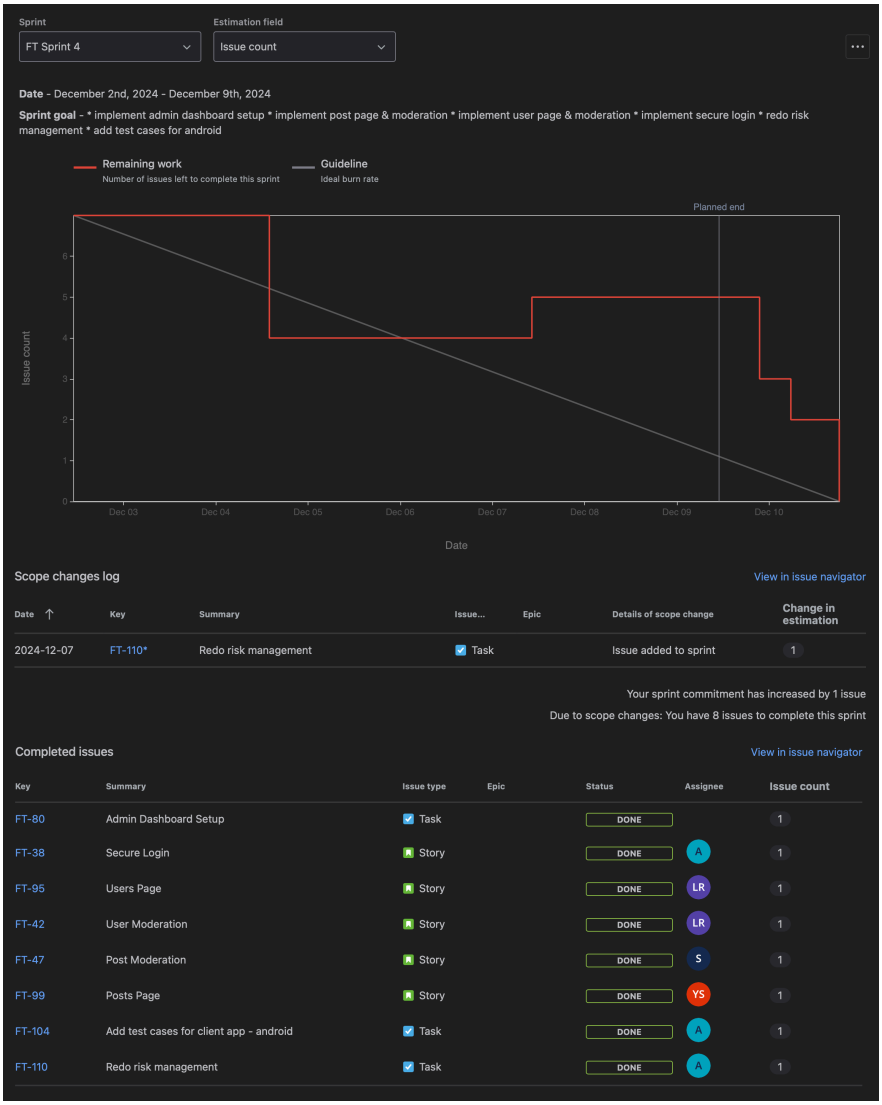


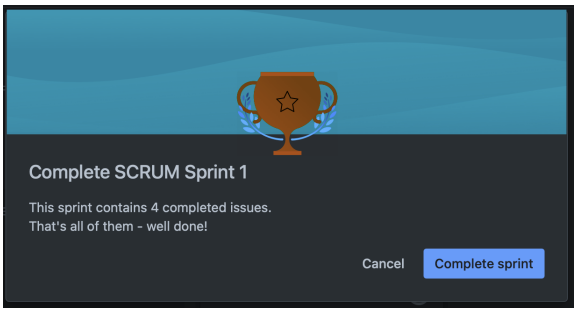
Figure 16: Sprint 4 report

As seen above, the fourth sprint covers Admin Dashboard Setup, Admin Secure Login, Admin Users Page, Admin User Moderation, Admin Posts Page, Admin Post Moderation, finalizing risk management log, and adding test cases for Android. All of these were completed on time and within the sprint duration. There is a case of the sprint ending one day later.

SPRINT RETROSPECTIVES

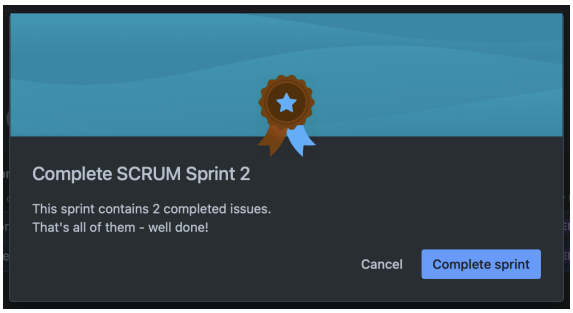
Sprint retrospectives are meetings held at the end of a sprint when in an Agile project. It is for identifying areas of improvement within the team and determining steps to enhance future sprints **(Scrum.org, 2019b)**.

Below are the evidences of sprint retrospectives of FeelTok:

SPRINT NUMBER	**RETROSPECTIVE**
Sprint 1  	**WHAT WENT WELL** ● Project setup (Firebase database connection and expo) on all team member's devices was implemented. ● User registration and login were implemented and tested successfully. This includes password validation, password handling,g and storing credentials to Firebase.

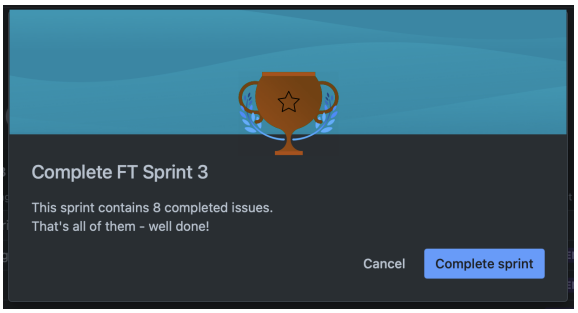
	WHAT DIDN'T GO WELL ● Used incorrect Firebase SDK in implementing user registration and login. **WHAT COULD'VE BEEN IMPROVED** ● Team collaboration and time management could have been improved by each team member. ● Technical documentation should be implemented such as comments. **WHAT DO WE NEED TO AVOID** ● Avoid using the incorrect Firebase SDK.

Sprint 2	**WHAT WENT WELL** ● Created a draft for the home page with dummy posts and a notification button. Two-factor authentication was implemented and tested successfully via the use of an external library for sending the OTP to the user's email inbox. ● Created a risk management log. **WHAT DIDN'T GO WELL** ● Implementing two-factor authentication was impossible through Firebase since it required a paid plan. ● Implementing Google Login was unsuccessful as it requires a development build via native components. Google Login will be moved to the next sprint.

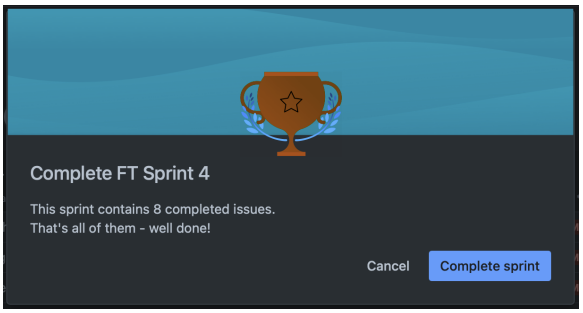


	WHAT COULD'VE BEEN IMPROVED ● Team collaboration and time management could have been improved by each team member. **WHAT DO WE NEED TO AVOID** ● Avoid Expo in Go mode and go for development build mode.

Sprint 3	**WHAT WENT WELL** ● Implemented all the necessary features such as Google Login, posting, profile page, and settings successfully. ● Implemented all features on time and within the sprint schedule. ● Team collaboration was good. ● Frequent daily stand-up meetings. **WHAT DIDN'T WENT WELL** ● A team member faced difficulty implementing Google Login due to technical errors. **WHAT COULD'VE BEEN IMPROVED** ● None **WHAT DO WE NEED TO AVOID** ● None



Sprint 4	**WHAT WENT WELL** ● Implemented all necessary features of the admin dashboard, such as post and user moderation. ● Implemented everything within the sprint schedule ● Finalized risk management log ● Finalized test cases ● Frequent daily stand-up meetings ● Team collaboration was good. **WHAT DIDN'T WENT WELL** ● None **WHAT COULD'VE BEEN IMPROVED** ● None **WHAT DO WE NEED TO AVOID** ● None



With the mentioned sprint retrospectives, we can reference the retrospectives in the next sprint and apply what can be improved, and most importantly what to avoid. This aligns each team member towards the project goal.

AFTER SPRINT RETROSPECTIVES

After completing sprint retrospectives, the team implements identified improvements for each sprint. The key focus is to have continuous improvements, along with increased team collaboration and team dynamics. Additionally, at the end of each sprint, testing is often executed and documented through test cases.

TESTING STRATEGIES

Testing strategies are approaches that outline how testing is performed during the development lifecycle. This is to ensure the quality, reliability, and performance of a software app. It defines the objectives, tools, and resources used for testing to identify and fix issues before release **(GeeksForGeeks, 2020)**.

P	Key	Name	Status
	SCRUM-T12	Create Post	APPROVED
	SCRUM-T9	Delete Account	APPROVED
	SCRUM-T13	Delete Post	APPROVED
	SCRUM-T8	Edit Credentials	APPROVED
	SCRUM-T4	Edit Profile	APPROVED
	SCRUM-T3	Email Login	APPROVED
	SCRUM-T2	Email Registration	APPROVED
	SCRUM-T7	Email Verification	APPROVED
	SCRUM-T14	Google Login	APPROVED
	SCRUM-T10	Report Post	APPROVED
	SCRUM-T1	Report User	APPROVED
	SCRUM-T5	Resend Email Verification	APPROVED
	SCRUM-T11	Sign Out	APPROVED
	SCRUM-T15	Timer	APPROVED
	SCRUM-T6	Two-Factor Authentication	APPROVED

Figure 17: Test cases in Zephyr Scale

FeelTok utilized the Zephyr Scale for creating test cases. The types of tests involve cross-platform testing, unit testing, integration testing, and system testing. Through this, the team identified bugs early on and we were able to conduct bug fixes right away.

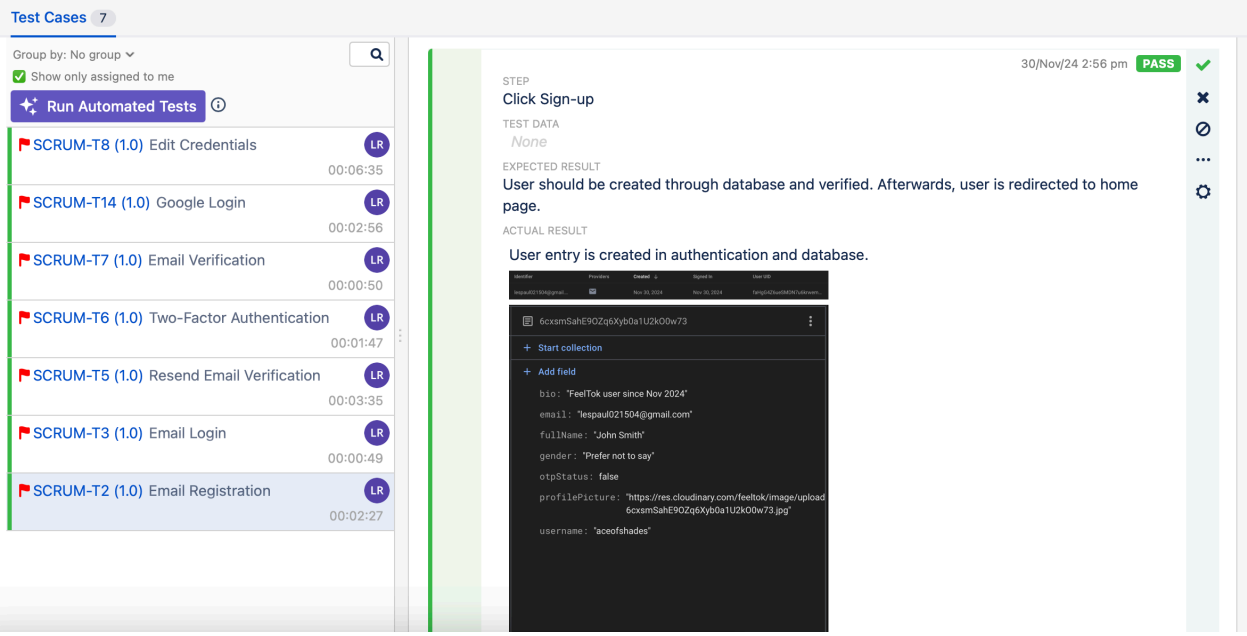


Figure 18: Email Registration test case

Unit testing was used for testing single functions such as email registration since it only utilizes one method within the Firebase API. As seen in the screenshot above, the test case is a pass, and a user entry is created through Firestore with the user's details.

Unit testing is then repeatedly executed for Sign-Out, Email Login, Report User, etc.

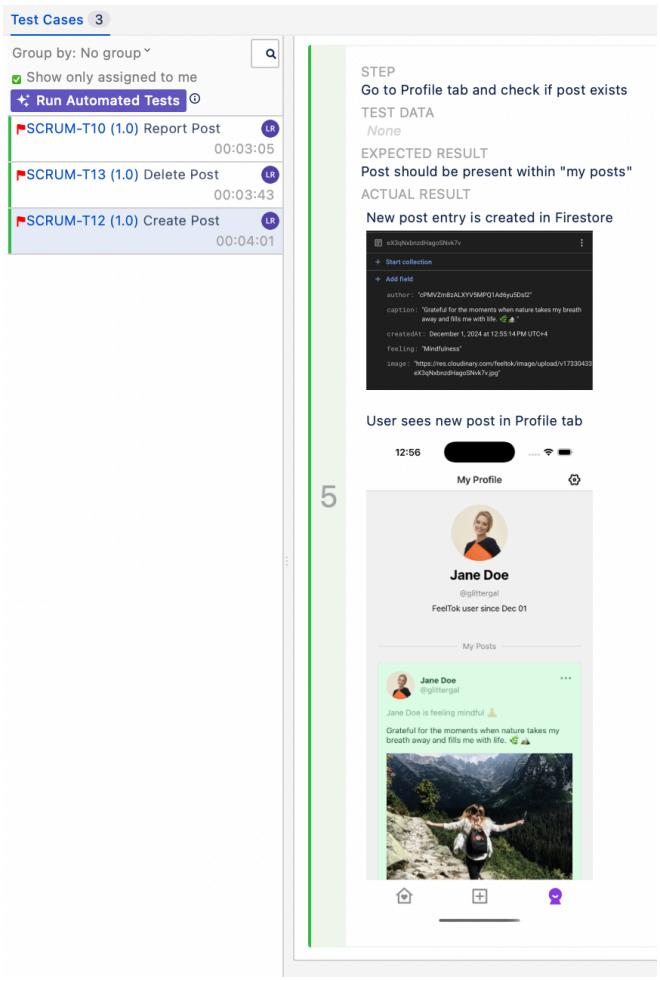


Figure 19: Create Post test case

Integration testing was conducted and is more towards how one function interacts with the rest of the system, an example of this would be the user creating a post, fetching said post's details, and displaying it on the user's profile page. As seen in the screenshot above, the test is a pass, and a post entry is created with post details such as images and captions. Within the profile page, it fetches the user's recently created post and displays it on the screen.

Integration testing is then repeatedly executed for the majority of the functionalities such as Delete Account, Edit Profile, Delete Post, Report Post, etc.

Key	Name	Progress	Status
SCRUM-R2	User Actions	100%	DONE
SCRUM-R1	User Authentication	100%	DONE
SCRUM-R3	User Posts	100%	DONE

Figure 20: Test cycles for iOS in Jira

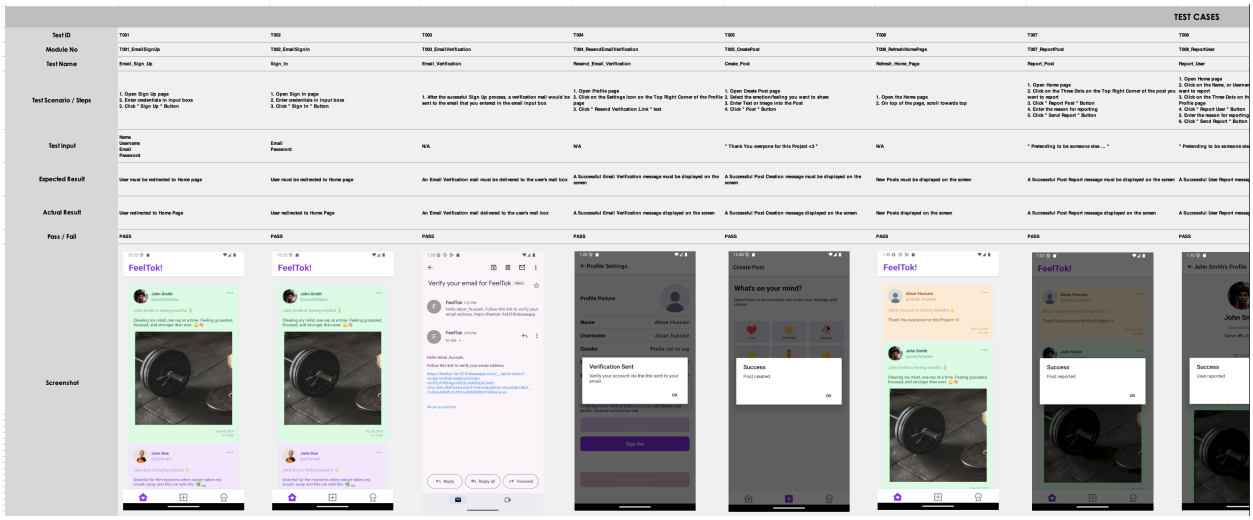


Figure 21: Integration test cases for Android in Excel

Lastly, system testing was conducted to test the entire app with all its functionalities implemented. This also includes cross-platform testing wherein the entire app is tested on Android and iOS. As seen in the screenshot, all three test cycles for iOS, categorized in their area, are all done and passed. Additionally, integration test cases for Android are all passed as well. At present, FeelTok is functional with all its features on both platforms with no errors.

CONCLUSION

The development of FeelTok through the Agile methodology has provided insights into both technical and collaborative aspects when developing software apps. The iterative process allowed the team to make continuous progress and improvements, adapt to changes and challenges, and deliver features that meet the requirements and project goals.

Throughout the sprints, we were able to prioritize important features such as user authentication, text and image posting, and admin moderation, ensuring the app's core functionalities were added before moving on to more complex features. Sprint planning, daily stand-up meetings, and retrospectives were facilitated to ensure smooth communication and problem-solving within the team.

Despite challenges relating to team collaboration, the project demonstrated clear communication and effective task management. The completion of FeelTok set the stage for future iterations and improvements. This just proves that Agile methodology is as effective when it comes to project management and team management.

If I were to give my opinion on the current app, I think the overall foundation such as design and current features is solid. However, user-to-user interaction is limited such as chatting and having a friends list. Additionally, user-to-post interaction is also limited wherein likes and comments are missing.

REFERENCES

Asana (2024). ***Burndown Chart: What it is and How to Use it (With Example) • Asana***. [online] Asana. Available at: https://asana.com/resources/burndown-chart.

Geekbot (2020). ***Daily Standup Meetings: Everything You Need to Know (Standup Agenda, Purpose, Common Pitfalls, and More!)***. [online] Geekbot blog. Available at: https://geekbot.com/blog/daily-standup-meeting/.

GeeksForGeeks (2020). ***Software Testing Strategies***. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/software-testing-strategies/.

GeeksforGeeks (2024). ***Beginner's Guide to Create Design Strategy***. *GeeksforGeeks*. [online] doi:https://doi.org/10/7717/8025/2963/6608.

Gibson, K. (2023). ***What Is Risk Management & Why Is It Important?*** [online] Business Insights Blog. Available at: https://online.hbs.edu/blog/post/risk-management.

Knotts, B. (2024). ***What Are System Requirements: Understanding Hardware Specifications - thinglabs***. [online] Thinglabs.io. Available at: https://thinglabs.io/what-are-system-requirements-understanding-hardware-specifications.

Laoyan, S. (2024). ***What Is Agile Methodology? (A Beginner's Guide)***. [online] Asana. Available at: https://asana.com/resources/agile-methodology.

monday.com (2022). ***What is Agile release planning?*** [online] monday.com Blog. Available at: https://monday.com/blog/rnd/agile-release-planning/.

Scrum.org (2000). ***What is Sprint Planning?*** [online] Scrum.org. Available at: https://www.scrum.org/resources/what-is-sprint-planning.

Scrum.org (2019a). ***What is a Product Backlog?*** [online] Scrum.org. Available at: https://www.scrum.org/resources/what-is-a-product-backlog.

Scrum.org (2019b). ***What is a Sprint Retrospective?*** [online] Scrum.org. Available at: https://www.scrum.org/resources/what-is-a-sprint-retrospective.

Scrum.org (2024). ***What is a Sprint Review?*** [online] Scrum.org. Available at: https://www.scrum.org/resources/what-is-a-sprint-review.

Visual Paradigm (2019). ***What is User Story?*** [online] Visual-paradigm.com. Available at: https://www.visual-paradigm.com/guide/agile-software-development/what-is-user-story/.