Kevin Wong 32105132
Robyn Castro 37283141
Catherine Lee 32556136
Omar Mohammed 52712130
Rohini Goyal 10861145
Jane MacGillivray 19953141
Xingyu Tao 33610149

STREAM

'cuz we've all been there

Requirements and User Stories

Motivation / Opportunity

Everyone has experienced times when group work is unevenly distributed or team members are unaware of their level of contribution. In an effort to help avoid these conflicts, we are proposing project STREAM. This is a project and task management android application, primarily targeting students. It aims to provide students a platform that will help streamline group work and avoid group conflict, while building project management skills by facilitating project planning.

Even though there are many task management applications, they are not tailored towards students. STREAM will target student needs and help them have easy access for all project details. We plan to create simple solutions for communication, reminders, and scheduling.

Problem Statement

The problem of	Working in a group project effectively.
affects	Post secondary and high school students.
The impacts of which is	Unneeded stress, group tension, and missed tasks due to miscommunication.
A successful solution would be	One that allows teams to manage tasks and communicate efficiently and effectively.

Product position statement

For	STREAM aims to provide students a platform that will help streamline group work and avoid group conflict.

Who	Students working on group projects.
Our system	STREAM is an Android application, so it will be implemented using only software.
That	STREAM facilitates group by providing a task management system and streamlines the sharing of information amongst group members.
Unlike	Jira, Asana, Visual Studios TFS.
Our product	STREAM is designed particularly towards students and will rely heavily on in-app notifications. It aims to help streamline group work and avoid group conflict, while building project management skills by facilitating project planning.

Users
STREAM will be primarily targeted towards students working in group projects.

Feature List
A login system will be implemented to allow users to login and create accounts via Facebook or email.

Users will be grouped together by a project to form a team. A projects screen will display all projects of a user and will allow users to create new projects for their respective team. A task management system will allow students to keep track of team progress and what needs to be done. The task manager will include a to-do list, and by clicking on one of the tasks on the list, details of the task will be displayed and user can send notifications to remind users of deadlines. Users will be able to remind other team members of uncompleted tasks. The project's home page will display the progress of the team's overall progress and the user's progress as well as shows all tasks assigned to the user. On these tasks, user can mark them as completed or edit them.

A calendar platform to schedule meetings will allow for students to work around different schedules. Members can set up a meeting by adding a meeting time, meeting location and meeting description. Members can also send notifications to other members to remind them to attend the meeting.

A pinboard section will allow for important information to be easily published and accessed by other users. It is intended to be details of upcoming deadlines, contact information and other information to be kept in one place.

Constraints:
STREAM will be an Android application. The biggest challenges we see in creating this application is managing the database and coping with threading synchronization issues. Users will need a fast and secure way to get, post, put, and delete data to the shared cloud database, Firebase. The app also needs to handle concurrent requests to edit the same resource and safely avoid situations where changes made by one person is overridden and lost. We will update our application in real time when data is changed in Firebase. For example, when a user creates a pinboard message or tasks for the project, other users who are on the application and connected to WIFI will see the updated change instantly.

Google Firebase's database storage limit for a free account is 1GB . All features for STREAM will require storing data in the database. We will need to ensure that enough storage has been allocated for each feature, without going over.

Scope and Limitations
STREAM will include a task management system, a calendar and a board.
The task management system will allow for users to break up the project and better manage their time. The calendar will let users set up meetings and the board will let users "pin" useful information pertaining to the project. The task management system and board are our primary features, and a simple chat functionality will be added in later, if time permits. The simple chat functionality is currently in a separate GitHub branch named "chat" and works but the team does not have the time to integrate it and improve the functionality and user interface.
Our team discussed having a Google docs integration but decided, due to time constraints, it was not feasible. For our final product, we decided to remove the chat feature and the explanation is provided below under "Changes to Requirements Document".

Assumptions and Dependencies

STREAM's database will be created using Firebase, so we assume that Firebase will provide a feasible way to create the database that stores all of the data STREAM will need. Firebase will be handling email login as well. We assume that Firebase uses a secure method of verifying a user's identity and storing all login information. Facebook SDK will also be used to allow for user's to log in to STREAM easily. Android Studios will be primarily used to create the app, which uses Java, and JUnit will be used for testing.

Use Cases

Use Case 1: Authenticate User

Allows a user to authenticate himself or herself in the application

Primary Actor: The App User

Stakeholders and Interests:

- Project Manager: Need to add the user to a team project and are assured that the user accessing their project is a member of their group.

Main Success Scenario:

1. The User opens the application
2. The System requests for the user's username and password.
3. The User inputs their login credentials and presses the login button.
4. The System verifies that the information entered is correct and authenticates the user.
5. The User is directed to the app's main screen and given access to their data.

Extensions and Alternate Flows:

- Alternate Flow 1: User Logs in with Facebook
    1. If at 2, the User selects "Login with Facebook" instead of inputting their username and password
    2. The System checks if the User is logged into Facebook on their device or prompts them to log in to facebook
- Alternate Flow 2: Create an Account
    1. If at 2 and the User selects "Create an Account", direct the user to a page
    2. The User inputs their name, username, password, and email.
    3. The App sends an email and prompts the user to enter their activation code
    4. The User inputs their activation code.
- Alternative Flow 3: User Inputs Incorrect Login Information
    1. If at 4, and the system finds that the login information inputted is incorrect
    2. The System displays a message saying that the username or password is incorrect

Open Issues:

1. The users may not send enter the correct activation key. The app should remain on the activate account screen until the user terminates the app or returns to the main login screen.

Preconditions:

- User must have an account

Postconditions:

- The user will have an account
- The user will gain access to their projects and tasks within the app

Use Case 2: Create a Project

Allows user to create a project and invite members to a project.

Primary Actor: App User

Stakeholder and Interest:

- Project manager: needs to make sure all members are included

Main Success Scenario:

1. User clicks create a project
2. The App displays a pop-up dialog screen for user to input project details such as name, description
3. User enters project name
4. User enters STREAM username or email of all other project members
5. User clicks invite button
6. App closes the pop-up dialog screen
7. System creates the project and directs user to project home screen

Extensions and Alternative Flows:

- Extension 1: Failed to Add Members

If at 4 the user clicks invite but the list of members entered contains invalid emails or users. The System will generate pop-up will come up to alert them that they have failed to add some members to their project.

Open Issues: Not applicable

Preconditions:

- Have an account

Postconditions:

- Project has been created
- Other members have been invited via email or usernames

Use Case 3: Create Tasks
Primary Actor: App Users (Project team member)
Stakeholder and Interests:

- Project member: Needs to be able to create task and assign it to a member

Main Success Scenario:

1. User clicks on add Tasks button on Tasks screen of the app
2. System displays a pop-up dialog prompting user to input task details such as task name, description, due date and assignee
3. User inputs fields for task name, description, assignee
4. User clicks OK and system closes the pop-up dialog
5. System creates a new task for the project with the specified parameters
6. User redirected to Tasks screen where they can see an updated user interface with newly created task

Open Issues:

1. Should user be allowed to assign tasks after or during create a task?

Preconditions:

- User has signed up for an account
- Project member has created a project
- Project member has invited team members to join the team

Postconditions:

- A task will be created and inserted to database
- Task will be added to the complete list of tasks
- Task will appear on the Project Home page of the user it has been assigned to

Use Case 4: Posting to Board
Primary Actor: App User
Stakeholders and Interests:

- Members of the Project: Need to have access to the posted material

Postconditions:

- Upon success, uploaded material is pushed and saved to DB
- Every other user can now have access to the material

Main success scenario:

1. On success the user lands back on the updated Board page

Extensions and Alternative Flows:
Open Issues: Not applicable

Preconditions:

- User needs to be signed in to application
- User needs to be in a project

Use Case 5: Reminding other project members.
Primary Actor: App user.
Stakeholders and interests: Another user on the same team will be receiving a notification on his phone reminding him of a task assigned to him.
Main success scenario:

- The user opens the app.
- User selects a task that he would like to send a reminder for
- The App displays pop-up dialog screen
- User enters a notification message into textbox of the pop-up dialog screen
- User can cancel or send using the buttons on the pop-up dialog
- A second user to which the task was assigned receives the notification
- The second user can then view the task by clicking on the notification

Extensions and alternative flows.

- The user may have to first select the project after opening the app. This would happen if the user has more than one project associated with his account. This can be done in the projects screen which is accessed by the projects icon in the toolbar menu.

Preconditions:

- The user needs to be signed in.
- The user needs to be part of a project with more than one member.
- The project needs to have at least one task created, with a team member assigned to the task.

Postconditions:

- A team member receives a notification/reminder.

Use Case 6: Adding Stream Users to an Existing Project
Primary Actor: App user.
Stakeholders and interests: The user being added to the existing project, and users who already have access to that project.
Main success scenario:

1. The user selects a project and accesses the Team page
2. The user chooses to add a member to the project
3. The app creates a pop up dialogue for the user to enter the username of a Stream user
4. The user types the username of a valid user and presses “add”.
5. The user that was entered in the pop up dialogue is added to the project.
6. The app will display the newly added user among the project’s team members.

Extensions and alternative flows.

- Extension 1: Failed to Add Member
    1. If at 4 the user clicks “add” but the username provided does not belong to a valid Stream user.
    2. The System will generate a message that will alert the user that the provided username is invalid.

Preconditions:

- User is signed in
- User created a project to work on
- The user that wants to be added to the project has an account created

Postconditions:

- The added user can access and contribute to project they were added to.
- The other users of the project can assign tasks to the added user.

Non Functional Requirements

- Performance Requirements

    Since this is a mobile application, the app is expected to be responsive at all times. So if the app is fetching data updates, the user interface should still be able to respond to user input without lagging. This is why the database will be asynchronous, so that data is updated real time. The performance/speed of fetching the data is dependant on the user’s connection speed and coverage.

- Safety Requirements.

    There are no safety issues that could result in harm or damage directly as a consequence of using the app. There are however a few issues related to protecting the user’s data, which are discussed below under security requirements.

- Security Requirements.

    Security is a major concern for this project. Users will be providing information such as their emails, names and facebook profiles. They would be expecting this information to be protected, and not accessible to unauthorized people or even the developers of the application. Since this is a project management application, the users will also be relying on this app to manage their workflow and notify them of upcoming tasks or deadlines. This means that any loss in app data could result in confusion between project collaborators, and deadlines being missed. Therefore, the app should provide the users with standard security features that guarantee the protection of user data. This is implemented through Firebase’s own security functionality.

- Scalability Requirements.

  Student projects are not on the scope of a corporation, so very likely, our target users will not need to store large quantities of information per project. However, there is a possibility that our app will become popular amongst students resulting in a large user base. Users will be storing project information on our application, so we need to ensure that our database can accommodate project information and files for all users. The database we choose will therefore provide options to increase the amount of storage used by an app. In addition, users are relying on our app to work on projects more efficiently, so we must fetch their data in a timely manner, regardless of the number of users we have, to not waste their time.

- Software Quality Attributes.

  The app is expected to be reliable because teams are working at different times of the day. Therefore, the app should provide users with the same user experience and performance at any given time during the day. Since there are a lot of devices that run on Android with different screen resolutions and screen sizes, the user interface of this app should remain presentable on different Android devices with different screen sizes. Our team will ensure that the user interface of the app will work on the LG G4 and Samsung Galaxy S6. Since the development team consists of seven people, the source code should be simple, easy to understand and documented well so that developers can easily understand and test the code.

Changes to Requirements Document

Due to time constraints, our team decided to remove the chat functionality of our application. As it stands, users are able to message each other at the basic level but there are a few bugs and the UI does not look pleasant. The implementation of the chat feature will exist in one of the branches in our repository. We also removed Use Case 5 that deals with chat hence users cannot add information from chat to the pinboard. Additionally, the team decided to remove user roles such as Project Leader and Project member from the application due to time constraints. The application will simply be Project members who will be able to add tasks, add pins to board, create meetings with calendar and send notifications to other members. Having a distinct Project Leader role in the application did not seem important as we thought when we initially designed the document because other members probably want to create tasks and send notifications to other team members. Removing the roles and chat features frees up more time to implement other necessary features such as Projects and Task Management. Tasks will no longer display a progress bar. Instead, a Project's home page will be implemented to show the project's progress and the user's progress on the project.

Also, we felt that it was prudent to allow users to add team members to a project after the creation of a project, so we added a Team page within each project for this functionality to be fulfilled. This scenario is described in Use Case 6.