Requirements & Team Policies

This document defines the Fit Social project. It outlines the project's goals, requirements, and high-level implementation, including how the Fit Social team will operate to reach these goals.

Team Info & Policies

Team members

- Lawrence: in charge of backend and project architecture.
- Aaron: In charge of frontend and databases
- Matthew: In charge of both frontend and backend.
- Michael: In charge of backend
- Asher: In charge of frontend and management

Project Artifacts

- Github
- Proposal Presentation

Communication

- Slack channel: fully on topic, CSE 403 assignment focused discussion.
- Discord server: on topic for each channel e.g. `#backend` will be only to talk about the backend portion of the project.

Product Description

Fit Social is a primarily mobile social media platform that allows users to track, share, and compare their fitness progress. The ultimate goal of Fit Social is to bring people who have the same passion for fitness together and help them to gain motivation and information toward accomplishing their fitness goals.

Features

1. One of the features that we will be implementing is a progress-posting system that allows users to post their workout progress via photos or videos, and attach numbers (like the amount of weight lifted) that can be analyzed by the platform across other data that can provide useful content and information.
2. The platform will serve short form content strictly related to fitness. Unlike other social media platforms that distribute content, focusing on this area allows users to immerse themselves into fitness and will be much more likely to reach their goals if they consume only content that promotes working towards their goals.
3. Another feature is the fitness statistcs and tracking tools. Integrating with the (IOS and Android) Health API, allowing users to track progress in-app, and aggregating data and running queries across the user base allows users to get a unique but useful analysis of their progress and how to optimize for more.
4. We want to be a hub for fitness communities. This means we need to support social features like commenting, direct messaging, profile management, and anything else that allows for a more connected fitness user base.

Stretch Goals

1. In order to maintain order across the platform we could, given we have time, build automated moderation features that enforce the community guidelines. This can include flagging possibly misleading information or enforcing civility in comment sections.
2. Users want only relevant content in their feed. We can implement and automated recommendation system that aggregates user preference data to serve only the most desired content for each user.

Use Cases

Matthew: User feed interaction

1. Actors: User that wants to view feed

2. Triggers: User presses home feed button
3. Preconditions: User has an account and is logged in
4. Postconditions: User views home feed
5. List of steps:
   - User clicks home feed button
   - System collects user preference data
   - System sorts posts by relevancy to user
   - System serves 10 most relevant posts
   - System presents feed data to user
   - User views home feed
6. Variations:
   - User account has been blocked from another user's content, thus being unable to view some posts that they normally would be able to view
   - System cannot find relevant posts to show user in main feed, thus presenting posts based on other criteria like popularity
7. Exceptions:
   - User does not have account, thus is unable to access a home feed
   - User clicks login
   - User enters username and password
   - System invalidates given information
   - Sends user back to login

Asher: Comment Section

1. Actors: A user that wants to comment on a post
2. Triggers: User clicks "comment" on a post's comment section
3. Preconditions: User is logged in
4. Postconditions: A notification pops up for the post creator and the comment is shown below the post and pushed to the database
5. List of Steps:
   - User clicks "comment" on a post's comment section
   - User types and submits comment
   - Comment is analyzed by the automod system (if enabled by post creator)
   - Comment is added to the database
   - UI is updated with comment for all users in the comment section
   - Notification sent to post creator
6. Variations:
   - Post creator does not have notifications from comments enabled, so no notification is sent
   - Creator does not have automod enable, so comment is not checked other than for following community guidelines.
7. Exceptions:
   - Comment fails automod check and is rejected
   - Commenter is banned from commenting by a post creator and thus cannot comment

Michael: Allowing access to photos/camera

1. Actors: User that wants to post a picture
2. Triggers: allow the user to access their pictures or camera from their device
3. Preconditions: user is on the uploading photos screen
4. Postconditions:
   - Success: access to all photos and camera from mobile device
   - Failure: must go directly to settings for system to allow access to their photos and camera
5. List of steps:
   - User presses the photo upload button
   - User decides to either choose photos from their mobile device or take a picture
   - System asks permission to allow access to their photos
   - User clicks accept and not decline
   - System will access all of the photos from the user's mobile device

    - User clicks on the photo to upload
    - User will choose whether or not to edit their photo
6. Variations:
    - Access to the user's photos
    - Access to the user's camera on their mobile device
7. Exceptions: User clicks on the decline button or system does not have access to photos until allowing photos from settings

Aaron: Creating an account

1. Actors: A user that wants to create a profile on FitSocial.
2. Triggers: The user wants to make an account, see other posts and to post their own pictures.
3. Preconditions: The user has an email and is 16 years or older.
4. Postconditions: The user creates an account and is able to start friending others, posting their own pictures and comment on others.
5. List of steps:
    - The user clicks the sign up button
    - The user is asked for email and password
    - The system asks for a email and password verification for the account
    - The user inputs the email and password verification and is prompted towards another page of information.
    - The system asks for things such as body weight, height, interests and goals, etc.
    - The user inputs the information required
    - User confirms all the information before fully signing up.
6. Variations:
    - The user has already used an email for the account and is prompted to login
    - The user inputs the wrong data so the system alerts the user to try again.
    - The email verification code is wrong so the system prompts to try again.
7. Exceptions:

- Non-matching login information, where the user enters a username or password that does not match with system database of user information, thus being denied access to account
- No account, where the user enters a username and/or password that does not exist, thus being denied from access to account

Lawrence: Upload new fitness statistcs

1. Actors: User that has new data to upload
2. Triggers: User clicks on the statistics menu
3. Preconditions: User is logged in
4. Postconditions: Frontend and backend updated to reflect new user data
5. List of steps:
    - User chooses a statistic widget to append to listed in the stats page
    - User enters new data
    - User clicks submit
    - Database updated with new information
    - UI (graphs, displayed statistics) gets updated
6. Extensions: user may instead edit a preexisting field instead of append
7. Exceptions: user has no previously submitted data, so editing is not possible

Non-Functional Requirements

Privacy and security

Security principles will be applied to keep malicious actors from gaining access to information they aren't supposed to access. Privacy will be enforced and users can choose to block other users and hide their profile from the public.

User experience

Since this project is a social media platform that focuses on reaching as many people as possible, user experience is a priority. The user interface must follow design principles proven to improve UX and accessibility. By using libraries like bootstrap we can quickly deploy a high-UX interface.

Scalability

The code base will follow software engineering principles for readability, modularity, and extensibility. These principles, alongside documentation, will allow the project to be scalable even above the scope of this course.

External Requirements

1. We require that all known errors are handled accordingly (automatically if possible).
2. The product must be installable or usable through a URL/link. We will either put it on the app store or will have a downloadable link.
3. We will open source the code and provide instructions on how to develop on, build, and run from source.
4. We will keep the scope of the project within the range of 5 students and will scale up or down depending on how it goes through the quarter

Team Process Description

Software toolset

- Rust for backend. Rust has a very extensive ecosystem of libraries for web servers like Actix and database ORMs like Diesel. These tools are the fastest, competing and often beating similar libraries in languages like C++. Unlike C++, Rust's strong type system and compiler lowers the chance of runtime errors and guarantees no memory errors.

- SurrealDB for database. SurrealDB is a Rust-implemented serverless database that uses a SQL-like query language. This makes it easy to use for those familiar with SQL, and makes simple data retrieval (like user account info) fast on the frontend as it can return JSON directly from the database without needing to go through the backend. This way, the backend would only be used for data aggregation and computation, increasing speed and decreasing complexity.

- ReactJS for the frontend. React is the most popular Javascript framework giving multiple advantages. Not only will developer familiarity remove the need to learn a new technology, React's ecosystem of libraries (React bootstrap, for example) and community support make it easy to develop with. React works so well because it is component based, meaning components can be composed to create complex user interfaces quickly.

Team member roles

Team member roles are assigned based on technology familiarity and personal preferences in such a way that workloads are evenly split and distinct enough to allow developer intendance.

- Lawrence: in charge of backend, project architecture. Will design and build the backend and will define the high-level project architecture (i.e. the technologies and design principles used).
- Aaron: In charge of frontend and databases. Will design and build the frontend, mainly focusing on bringing database data to the frontend and tying that with the UI.
- Matthew: In charge of both frontend and backend. Will build on both the frontend and backend, getting the two to interact effectively.
- Michael: In charge of backend. Will build the backend making sure software design principles are followed.
- Asher: In charge of frontend and management. Will build and design the user interface. Also will make sure deadlines are met and team members are effectively contributing.

Team schedule

- Frontend:

  Every one to two weeks one of these features will be implemented in the following order.

  1. User login and profile page
  2. User stats page & content uploading
  3. Main feed page
  4. Comment section, mentions, and direct messaging pages
  5. Quality of life + UI enhancements

- Backend:

  Every one to two weeks one of these features will be implemented in the following order, meant to go along with the needs of the frontend.

  1. Database schemas, user authentication, and user data persistence
  2. User fitness data retrieval and augmentation and fitness statistcs
  3. Platform-wide content serving based on calculated user preferences
  4. Irrelivant content filtering, comment persistence, automated moderation

Risks

1. Too much new technology. The tech stack contains technology that is unfamiliar to some or all team members. While the challenge and learning experience is a reason for choosing these technologies, there is a world where becoming skilled enough to use the tech stack takes too long.
2. Too many features. To compete with other social media platforms and to make our project useful, we need to build features that usually takes professional teams a lot longer. We must choose which are the most necessary features and execute them well in order to finish on time.
3. Project infrastructure unsuitable for features. If we want to take a data analytics approach to providing user statistics, do automated moderation, serve content based on user preferences, or any other of the platform-defining features, we need to make sure the software we've built has the modularity to support these new features as we develop. If we lack the appropriate infrastructure on the frontend or backend we risk needing to rebuild large parts of the project to support these "lower level" features.

External feedback

External feedback from users will be very useful once we have the project-defining features like fitness statistics up and running. Other features, like short form content and user preferences already have proven to be a good idea, so those can come later down the line as we already know how a successful implementation looks like.