OMNIGYM APP

ITERATION THREE

CSC 4110: Software Engineering

Instructor Jason E. Myers

Wayne State University

April 26, 2025

Group 13: The Dream Team

Kliman Darawish   Viktor Gjorgevski   Abdulla Maruf

Jawad Rashid   Robert Simovski   Violet Yousif

Table of Contents

1. Omnigym App: Iteration 3

This document provides a comprehensive overview of our team's project, Omnigym Social App, showcasing the artifacts that demonstrate our development process, stakeholder involvement, and adherence to best practices in software engineering.

1.1 Proposal

Omnigym is a unique gym community-building social app that combines modern technology, user-friendly design, and high security standards to provide a seamless experience for both gym members and staff. Our concept addresses the needs of fitness enthusiasts and gym administrators alike, aiming to improve user engagement and increase profits for affiliated gyms. Our app offers gym members a platform to connect with others through messaging, friendly leaderboard competitions, shared fitness interests and routines, workout creation and tracking, and opportunities to celebrate each other's achievements—among many other features. Additionally, by incorporating member profiles, gym trainers now have a clear way to identify members and understand their fitness goals. This enhancement improves their ability to connect with members and boost sales for both the trainers and the gym.

Furthermore, the gym benefits by fostering a community where members feel more comfortable interacting with one another, participating in gym-related events and activities, and remaining loyal after forming meaningful relationships.

1.2 Why Choose Our Team?

- **Technical Excellence:** We leveraged a full-stack development approach using React Native + Expo for the frontend, Django for the backend, and Supabase/PostgreSQL for data management.
- **Scalability & Security:** Omnigym follows industry best practices for data protection, authentication security, and performance optimization to support a growing user base.
- **User-Centric Design:** Our UI/UX design is based on extensive user research and advertising strategies like color theory, ensuring an intuitive and attractive experience tailored to gym members and trainers.
- **Agile Development Process:** We implemented an iterative development approach using Scrum methodology, ensuring rapid feature deployment and continuous improvement.
- **Future Vision:** Our roadmap includes AI-powered fitness recommendations, a social networking feature, and real-time workout tracking to enhance engagement and personalization.

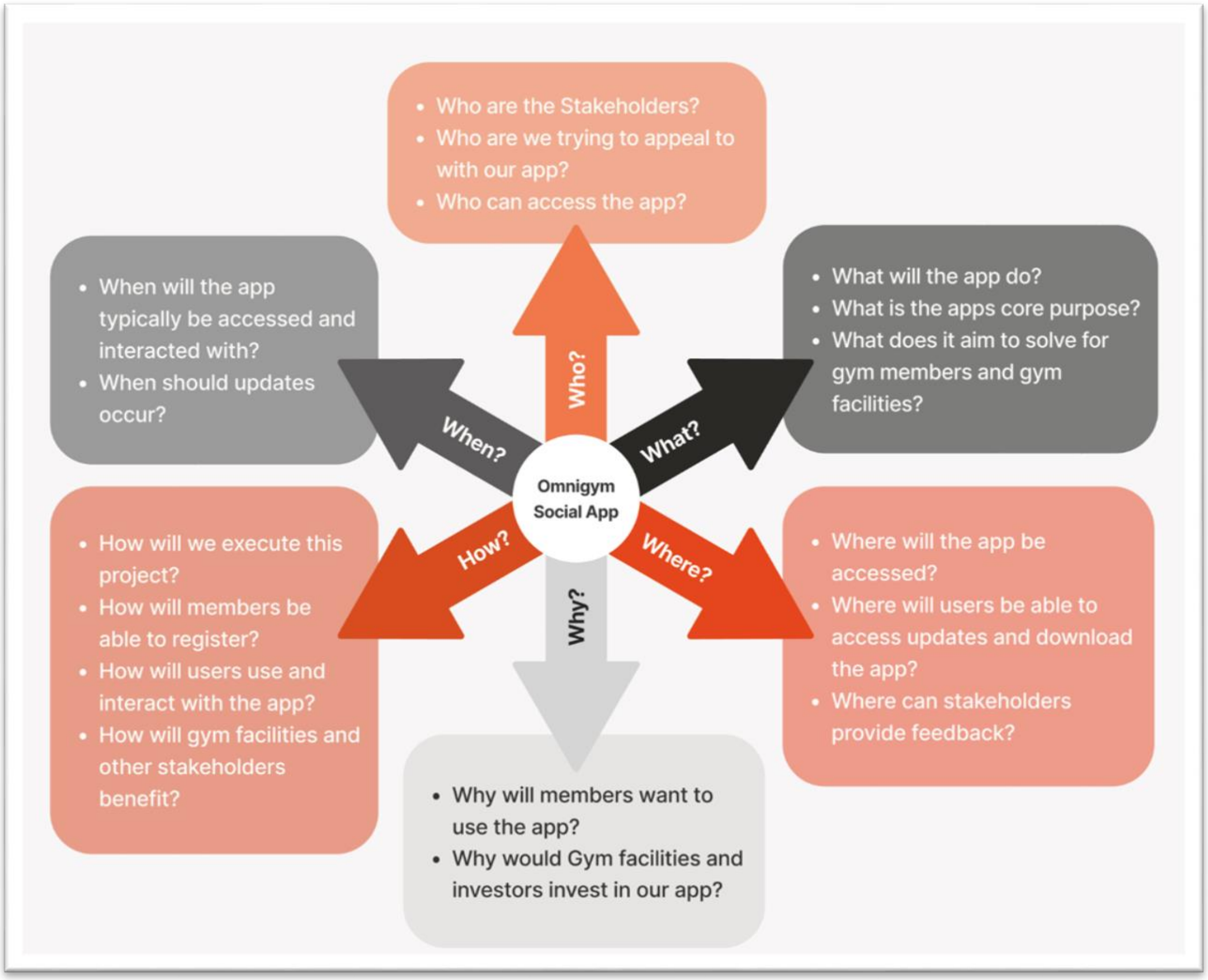
By selecting our team, you are choosing a dedicated, innovative, and well-rounded group of developers committed to delivering a high-quality, scalable, and user-friendly gym social app.

2. Project Artifacts

2.1 Idea Generation Final Draft

The final idea generation came from a process. We first started by playing the game two truths and a lie. After guessing each other's and getting to know each other we all had a common interest in to workout. This was not the final decision marker of our project idea but put us one step closer. After this we did our brainstorming and while presenting ideas, we all had a similar idea of a gym app, so we based it off of that app, implementing each of our ideas until we got our final draft of our idea. We used Edotor as a way to track our ideas and features. The revised diagram can be seen in Appendix A at the end of the document.

2.2 Starburst Accord


Omnigym Social App
Who?
- Who are the Stakeholders?
- Who are we trying to appeal to with our app?
- Who can access the app?
What?
- What will the app do?
- What is the apps core purpose?
- What does it aim to solve for gym members and gym facilities?
Where?
- Where will the app be accessed?
- Where will users be able to access updates and download the app?
- Where can stakeholders provide feedback?
Why?
- Why will members want to use the app?
- Why would Gym facilities and investors invest in our app?
How?
- How will we execute this project?
- How will members be able to register?
- How will users use and interact with the app?
- How will gym facilities and other stakeholders benefit?
When?
- When will the app typically be accessed and interacted with?
- When should updates occur?

2.3 Development Model

We adopted an Agile-Scrum development model to ensure that each task is assigned to the appropriate team member. The model that each of the tasks laid out from backend to front end. The backend consists of Supabase and Django while the front end is React Native + Expo (Supabase). With the structure set forth we were able to get everyone assigned to certain spots that they are proficient in allowing a smooth process.

2.4 Requirements Matrix

ID	Requirement	Type	Related Features	Priority	Validation
Fun-01	Users' registration, login, and profile creation	Functional	Profile Modules	High	Unit Testing
Fun-02	Setup routing for all user flows	Functional	Navigation/Routine	High	Scenario Testing
Fun-03	Make creation of groups and events	Functional	Event Management Module	Medium	Scenario Testing
Fun-04	Implement full Supabase integration (register, forgot password)	Functional	Navigation/Routing, Authentication	High	Scenario Testing
Fun-05	Allow customizable profile editing (bio, profile picture, settings)	Functional	Profile Modules	High	Scenario Testing
Fun-06	Enable users to log workouts and track personal records	Functional	Workout Tracking	High	Scenario Testing
Fun-07	Setup role-based access (member, trainer, admin)	Functional	Authentication, Admin Panel	High	Unit Testing
Fun-08	Display leaderboards for PRs and fitness progress	Functional	Competition Module	Medium	Scenario Testing
Fun-09	Enable messaging between members and trainers	Functional	Messaging Module	Medium	Usability Testing
Fun-10	Enable group and event creation with member participation	Functional	Event Management	Medium	Scenario Testing
Fun-11	AI-based workout recommendations (future feature)	Functional	Workout Tracking	Low	Unit Testing
Non-01	Deliver a responsive and intuitive user interface	Non-Functional	Frontend (Native React)	High	Usability Testing

Non-02	Ensure secure handling of user data (authentication, encryption)	Non-Functional	Backend (Supabase, Security Modules)	High	Penetration Testing
Non-03	Ensure scalable backend to support growing user base	Non-Functional	Backend (Supabase, Firebase)	High	Load Testing
Non-04	Provide smooth performance on both iOS and Android	Non-Functional	Frontend Optimization	High	Performance Testing
Non-05	Optimize app load time (under 3 seconds for core screens)	Non-Functional	Frontend Optimization	Medium	Load Testing
Non-06	Ensure offline data resilience for local workout logging	Non-Functional	Local Storage	Medium	Scenario Testing
Non-07	Maintain simple, clean navigation across all modules	Non-Functional	Navigation/Routing	High	Usability Testing
Non-08	Ensure daily active use is encouraged via notifications	Non-Functional	Notifications Module	Medium	Usability Testing
Non-09	Ensure easy maintainability and future feature expansion	Non-Functional	Backend Codebase	Medium	Code Review
Non-10	Guarantee GDPR compliance for user data privacy (future enhancement)	Non-Functional	Data Security	Low	Legal Review

2.5 Stakeholders

Key stakeholders include:

- **Gyms:** Gym owners/companies
- **Primary Users:** Community members at each gym and people seeking workout buddies, fitness advice, workout guidance, and motivation.
- **Personal Trainers:** Depending on the gym, some gyms' staff trainers and others allow outside professional's access to their facilities. To connect and interact with gym members.
- **Gym Staff:** Gym employees, managers, admins, and commissioned trainers.
- **Investors:** To provide funding for the app maintenance and its staff.
- **App Staff and Support Team:** For customer service and programming updates.

2.6 Stakeholder Requirements Matrix

ID	Requirement	Stakeholders Benefiting	Notes
Fun-01	Users' registration, login, and profile creation	Gym Members, Gym Staff, Development Team	Member access control, admin verification process
Fun-02	Setup routing for all user flows	Gym Members, Personal Trainers	Smooth app navigation critical for all users
Fun-03	Make creation of groups and events	Gym Members, Gym Staff	Community building and event management
Fun-04	Implement full Supabase integration (register, forgot password)	Development Team	Secure and scalable backend foundation
Fun-05	Allow customizable profile editing (bio, profile picture, settings)	Gym Members	Personalization increases daily engagement
Fun-06	Enable users to log workouts and track personal records	Gym Members, Personal Trainers	Tracking progress, trainer accountability

Fun-07	Setup role-based access (member, trainer, admin)	Gym Staff, Personal Trainers, Gym Owners	Proper access control, operational clarity
Fun-08	Display leaderboards for PRs and fitness progress	Gym Members, Gym Owners	Motivation for users, showcases gym success
Fun-09	Enable messaging between members and trainers	Gym Members, Personal Trainers	Direct communication, trainer-client support
Fun-10	Enable group and event creation with member participation	Gym Members, Gym Staff	Promotes community and gym event engagement
Fun-11	AI-based workout recommendations (future feature)	Gym Members, Personal Trainers	Personalized programs, competitive advantage
Non-01	Deliver a responsive and intuitive user interface	Gym Members, Personal Trainers	User retention, app satisfaction
Non-02	Ensure secure handling of user data (authentication, encryption)	Gym Members, Gym Staff, Gym Owners	Privacy compliance, legal protection
Non-03	Ensure scalable backend to support growing user base	Development Team, Gym Owners	Future-proofing app as gym network grows
Non-04	Provide smooth performance on both iOS and Android	Gym Members, Personal Trainers	Accessibility regardless of device
Non-05	Optimize app load time (under 3 seconds for core screens)	Gym Members, Personal Trainers	Reduces bounce rate, improves UX
Non-06	Ensure offline data resilience for local workout logging	Gym Members	Access even with poor gym WiFi
Non-07	Maintain simple, clean navigation across all modules	Gym Members, Personal Trainers	Reduces user friction, improves adoption
Non-08	Ensure daily active use is encouraged via notifications	Gym Members, Gym Staff	Reminders, events, retention marketing
Non-09	Ensure easy maintainability and future feature expansion	Development Team	Easier future updates and fixes
Non-10	Guarantee GDPR compliance for user data privacy (future enhancement)	Gym Members, Gym Owners	Trust and legal adherence

Stakeholder	Key Requirements	Why It Matters
Gym Members	Fun-01, Fun-02, Fun-03, Fun-05, Fun-06, Fun-08, Fun-09, Fun-10, Fun-11, Non-01, Non-02, Non-04, Non-05, Non-06, Non-07, Non-08	Core users; want easy login, workout tracking, leaderboards, communication, events, app speed, and security.
Personal Trainers	Fun-02, Fun-06, Fun-07, Fun-09, Fun-11, Non-01, Non-04, Non-05, Non-07	Need smooth app use, client data access, messaging with clients, and tools for workout planning.
Gym Staff	Fun-01, Fun-03, Fun-07, Fun-10, Non-02, Non-08	Need to manage users, verify memberships, oversee events, ensure data security, and engage members via notifications.
Gym Owners	Fun-07, Fun-08, Non-02, Non-03, Non-10	Focused on retention, scaling operations, protecting data privacy, and showcasing gym performance via leaderboards.
Development/Support Team	Fun-04, Non-03, Non-09	Focused on building scalable, maintainable backend, authentication flows, and smooth app updates for future features.

Most requirements overlap between Gym Members and Trainers, reflecting our app's community-driven design philosophy. However, scalability and backend security remain critical non-functional priorities for Gym Owners and the Development Team to ensure long-term app success.

2.6 Toolsets Created

Design and Database Management Tools:

Figma – Used for UI/UX design and prototyping.

Edotor – Used for idea generation during initial concept creation, both for the app concept and for the pages and features we wanted implemented in the app.

Supabase– An open-source database providing PostgreSQL, user authentication, file storage, image storage, statistical overviews and graphs, etc.

Draw.io (diagrams.net) – Used to create UML diagrams for visualizing system architecture, workflows, and relationships between components.

Frontend:

React Native + Expo – Used for creating native mobile apps using HTML, CSS, JavsScript and React. React Native enables cross-platform development with a single codebase, and Expo simplifies the process with tools for building, testing, and deploying apps faster.

Backend:

Django (Python) – Chosen as the web framework for handling API requests, managing backend logic, and integrating with the database to power the application's server-side functionality.

Database:

Supabase – Used as the backend-as-a-service platform for managing the PostgreSQL database handling authentication, and providing real-time data and API access for applications

Development & Collaboration Tools:

Microsoft Teams – Used for team communication and collaboration.

GitHub – Version control system for tracking and managing code changes.

Excel Gantt Charts – Utilized for project scheduling and progress tracking.

MS Planner – Used for task management and team coordination.

Online Scrum Form Template – This is the form the scrum master used to report meeting tasks, completions, and goals as a team. The online scrum template can be found on the *following page*.

WEEK

Client:	-- NULL --

Product Owners:	The Dream Team (Group 13)	**Scrum Master:**	Kliman Darawish

Sprint:	Date:	MM/DD/YYYY	Time:	00 hr 00 min
Last Sprint:	Date:	MM/DD/YYYY	Time:	00 hr 00 min
Attendees:	Kliman Darawish		Abdulla Maruf	
	Robert Simovski		Jawad Rashid	
	Viktor Gjorgjevski		Violet Yousif	
Achievements:				

Product Backlog: (In-progress tasks and by whom, forms to be created)
<<<<<state GP1 goals here>>>>>
state in-progress tasks here and by whom
form creation, description

Sprint Goals: (New requirements/events, existing goals)
new requirements / events, if any
existing goals: front end, application coding, cc, pres, written

Team Availability: (Tasks and research given to members on individual time)
<<Example: John James will research Design Ideas for Sales Websites>>

Assign Backlog Items: (Assigned tasks pertaining to completion of Backlog goal)
<<Example: Suzie Q will work on Front End - reqt 1e>>

Retrospective: (What went right? What went wrong?)
What went right?
What went wrong?

2.7 Predicted Process Model

Agile Development:

Because of its adaptability, user-centric methodology, and capacity to change to meet the changing demands of trainers and gym clients, we decided to employ the Agile Model for our project. Agile makes incremental delivery possible, so we can release important features like fitness monitoring and gym scheduling early while still making constant improvements to the app.

Important advantages of this strategy include:

- Regular sprint reviews guarantee alignment with user needs and eliminate unnecessary features. This promotes frequent feedback and collaboration.
- Risk Mitigation: We identify and fix problems early by testing and improving features every sprint.
- Faster Time to Market: We can deliver value to users sooner by delivering in small increments.
- Continuous Improvement: Over time, we are able to improve the app and our development process thanks to Agile's iterative nature.
- Adaptability to Industry Trends: Agile allows us to incorporate new features, such as social challenges, as they appear in the ever-changing fitness sector.
- Agile guarantees that our Fitness Leaderboard App stays useful, effective, and in line with user expectations by placing a strong emphasis on transparency, quality assurance, and reactivity.

2.8 Scrum/Sprint Records and Meeting Minutes:

For easier viewability and organization, the weekly scrum meeting records can be found in the Appendix C section located at the end of the documentation.

2.9 Process Framework

Our project follows an **Agile-Scrum framework** integrated with CI/CD and automated testing to ensure a smooth and efficient development process.

Scrum-Agile Principles:

Our main development methodology is Agile-Scrum, with a focus on:

- Iterative Development: Work is organized into sprints, enabling incremental progress and continuous improvement.
- Regular Stakeholder Feedback: Consistent sprint reviews guarantee that the application adapts to user requirements.
- Cross-Functional Collaboration: To produce high-quality features, developers, designers, and stakeholders collaborate closely.
- Flexibility and Adaptability: The app's relevance can be maintained by adjusting priorities in response to user feedback.

CI/CD Implementation (Continuous Integration & Continuous Deployment):

To streamline development and deployment, we integrate CI/CD pipelines, ensuring that:

- Code changes are automatically tested and merged, reducing integration issues.
- New features and bug fixes are deployed quickly, keeping the app updated.
- Development remains agile, allowing for frequent and stable releases.

Automated Validation and Testing

To ensure dependability and code quality, we used automated testing:

- Unit testing makes sure that each component works as it should.
- Integration testing to confirm that several modules worked together seamlessly.
- End-to-end testing to prevent any problems by simulating actual user interactions.
- Performance testing to evaluate how responsive and stable the application is when loaded.

Tooling and Workflow Support:

- GitHub Projects: Used to manage sprint tasks and assign ownership.
- MS Teams + MS Planner: For team coordination, daily updates, and sprint planning.
- Supabase CI Hooks: Used for database update testing and deployment staging.
- Expo CLI & React Native Debugging Tools: Supported frontend testing in development mode.
- GitHub Actions (optional): For CI/CD automation workflows and build verification.

Sprint Cadence and Team Rituals:

- Sprint Planning: To assign tasks and establish priorities, every Monday.
- Daily Check-ins: Microsoft Teams asynchronous updates.
- Sprint Retrospectives: Held right after reviews to evaluate and pinpoint areas for improvement.

Process Challenges and Adaptations:

The team had trouble with variable local settings and merge conflicts in the early sprints. Consequently, we developed backup branches prior to significant merges, standardized environments using.env templates, and provided Git training videos.

Versioning Strategy:

To identify stable milestones, code was iteration-tagged (e.g., v1.0, v2.1-beta) using GitHub Releases.

As a result, the team was able to maintain consistent deployment references and roll back if necessary.

Agile-Scrum, CI/CD, and automated testing are all combined in our process framework to guarantee a flexible yet disciplined development workflow. This method allows for high-quality software that adapts to the changing needs of trainers and gym consumers, as well as quick feature creation and ongoing improvement.

2.10 Document Control

Document Control Tools			
Tools:	**Purpose:**	**Managed by:**	**Location:**
Figma	Prototype Designs	Violet	Connected to Teams App Navbar
MS Planner	Task Creation. Check off fulfilled tasks. Setup collaboration between members.	Jawad and Violet	Integrated into Teams
GitHub	Merge Messages, Discussion topics, version control	Violet	OmniGymSocialApp Repository
SharePoint	Documents and Projects	Jawad	Microsoft Office Shared folders
Microsoft Teams	Pinned Chats for due dates, references, documents, meeting calls, etc.	Jawad and Violet	Sidebar icon shows pinned files and comments. Some apps can be located in top Navbar of Teams.
Violet's GitHub Video Shorts	Tutorials on using Git with GitHub	Violet	Shared on Teams. Can locate in shared files and pinned comments.
Visual Studio Code + GitHub Desktop	Version control, branch creations, code documentation	Jawad and Violet	Individual branches using VSCode + GitHub and Git. Version Control with GitHub. Purpose stated for each VSCode file in code.

Primary Document Control Files		
Iter. #	**Document Name:**	**Location:**
1	Brand Designs	GitHub Brand Folder. MS Teams files.
1	Project/Iteration 1	Microsoft SharePoint Folder
1	Edotor Diagram	Project Management Excel Doc in SharePoint Folder, or Navbar on MS Teams
1	Omnigym Project Pre-Release 1	GitHub Pre-Release list Versions: v0.1.1-alpha, v0.1.2-alpha
1, 2	Meeting Minutes	Microsoft SharePoint Folder
2	Quality Manual	Microsoft SharePoint Folder
2	Project/Iteration 2	Microsoft Office Shared folders
2	Omnigym Project Pre-Release 2	GitHub Pre-Release list. Versions: v0.1.3-alpha, v0.1.4-alpha

2.11 Waste Log

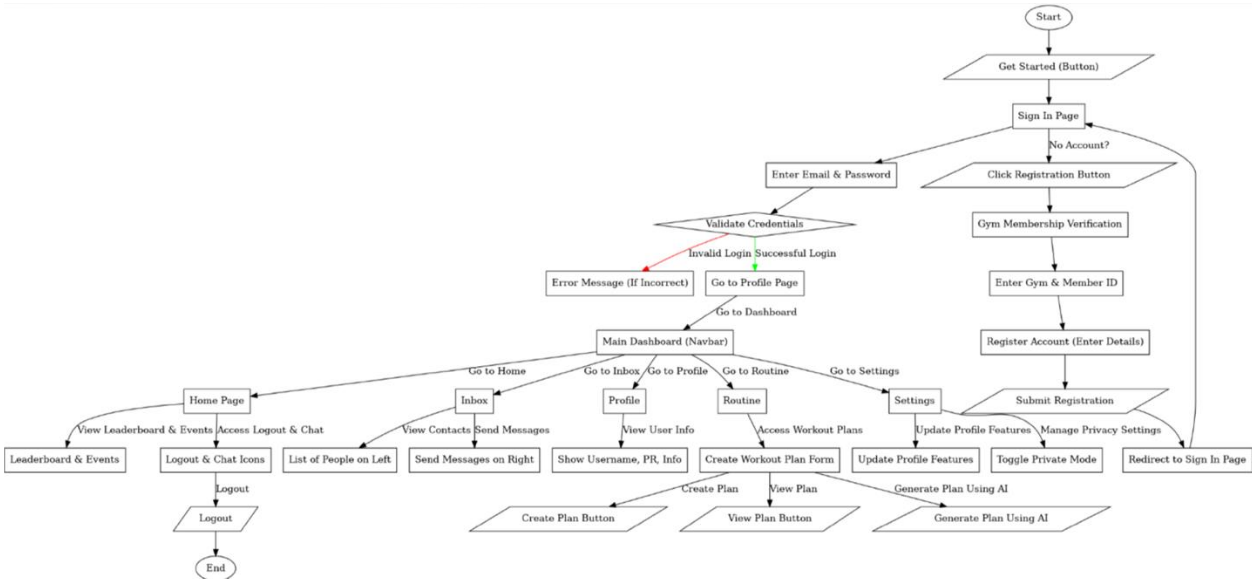
Waste Type			
Date	**Description of Waste**	**Impact**	**Action Taken or Suggested Fix**
Frontend			
3/5/2025	Rebuilt login screen layout 2 times	5 hours lost	Finalized mockups before coding
3/7/2025	Misused useEffect causing infinite loop	App crash, 1 day delay	Added correct dependency array
3/10/2025	Styles inconsistent due to manual inline styles	Time wasted fixing styles	Adopted centralized theme with StyleSheet
3/12/2025	Assets not optimized (e.g., large images)	Slow performance	Compressed images and used Image.prefetch
3/14/2025	Wrong navigation stack used (Stack vs Tab)	Had to refactor entire nav	Reviewed React Navigation docs beforehand
3/16/2025	Overuse of re-renders due to improper state usage	Laggy screens, 2 hrs debugging	Used useMemo and React.memo where needed
Backend			
3/6/2025	Forgetting to set Supabase row-level security (RLS)	Exposed data temporarily	Created default RLS policy template
3/8/2025	Misconfigured Supabase storage bucket permissions	Uploads failed	Reviewed and documented permission setup
3/11/2025	Using too many rpc() calls for simple queries	Overcomplicated backend	Replaced with direct .select() queries
3/13/2025	Schema changes broke existing client-side logic	Caused app crash	Set up versioning for table schemas
3/15/2025	Delayed debugging due to missing Supabase logs	3 hrs wasted	Enabled full query and auth logging

2.12 Quality Policies

In Omnigym, our development quality policy revolves around the user, where every line of code and every architectural decision contributes directly to the user experience. Through the application of bleeding-edge technology such as React Native for a responsive mobile experience, Django for robust backend support, and Supabase for optimal database management, we build an app that is friendly to use and full of features. Our commitment to security never wavers utilizing strong encryption protocols and conducting regular security audits guarantees user data is always safe and in keeping with industry best practices. Our infrastructure is likewise scalability and performance-optimized, guaranteeing that as our community grows, the app will remain fast, responsive, and trustworthy.

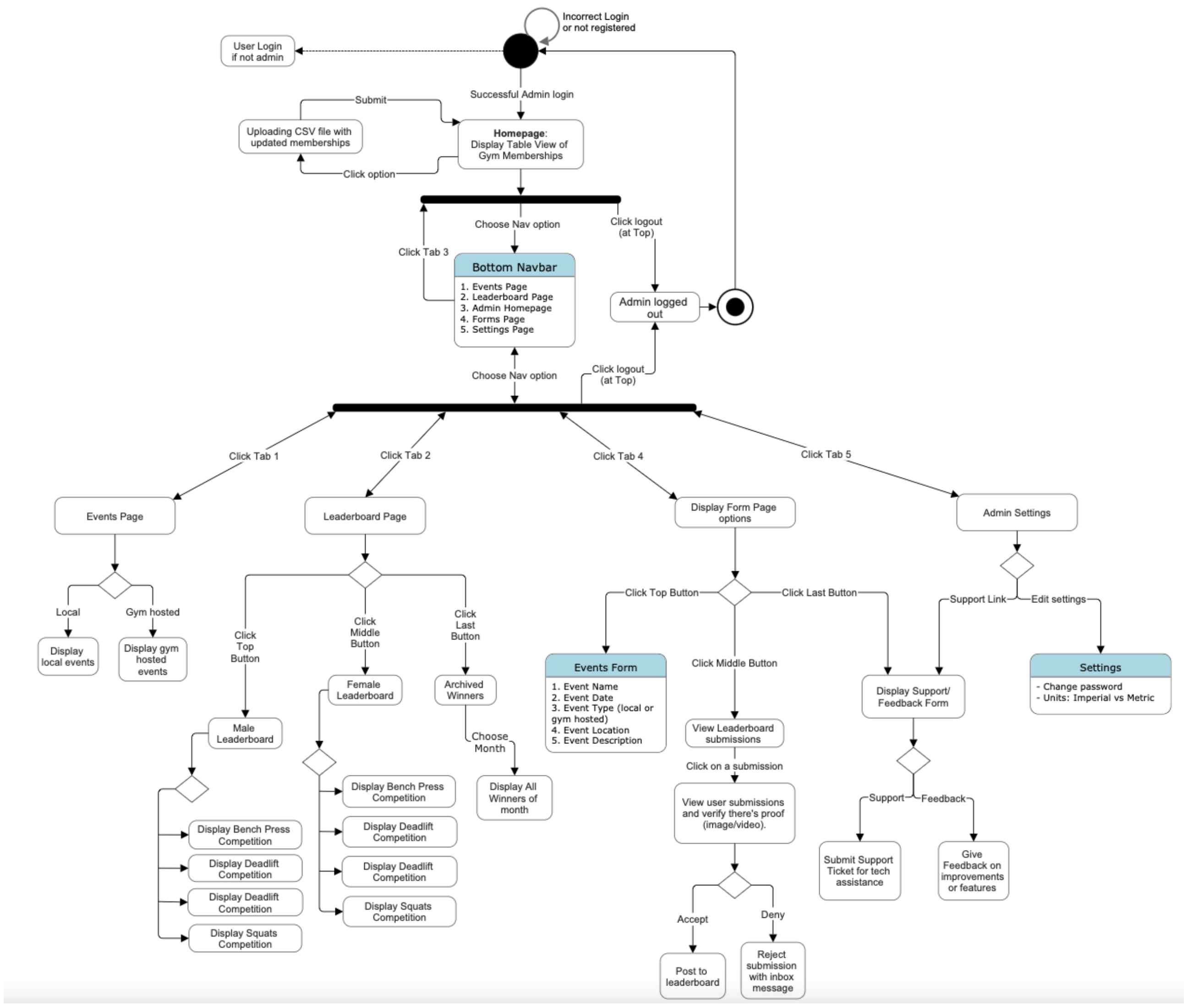
We also focus on agile development practices and full transparency throughout our project life cycle. Our Scrum cycle allows us to iterate fast against stakeholder feedback, enhance functionalities, and maintain documentation clean at every step. This not only creates effective communication among end-users, stakeholders, and developers but also establishes trust and accountability. In the future, Omni gym keeps aiming to innovate and develop by venturing into edge-cutting features such as integration of AI and real-time monitoring, further enhancing the user interface and offering novel forms of engagement with the community at the gym. Through continued incorporation of feedback from users, we ensure that the app grows according to the fitness enthusiasts and the gym attendants' specifications, ultimately having a dynamic, interactive, and safe community platform.

2.13 Activity Diagram



The activity diagram is a visual representation of the user flow within the gym membership application. From the activity diagram (orange chart) in Appendix A, you can see an overview of the user input and pages. The more detailed diagram (above) provides a more functional view of the main navigation paths with buttons. The process starts with users opening "Get Started" and logging in or registering a new account, including gym membership verification. When the login credentials are input, they are authenticated by the system and allowed into the profile/dashboard or display an error message when incorrect. The main dashboard is made up of navigation elements for the home page, inbox, profile, workouts, and settings, where users can create and view workout plans. Along with this, the users can also communicate through the inbox by viewing contacts and messaging and attending and viewing events through the leaderboard and events menu on the home page.

The next activity diagram (below this section) is for the admin pages. It follows the navigation and functional specifications that a registered admin would follow when logging in to the app with their admin-provided email. Admins cannot register on their own. The admin emails are provided by the Omnigym team and given permissions through Supabase.



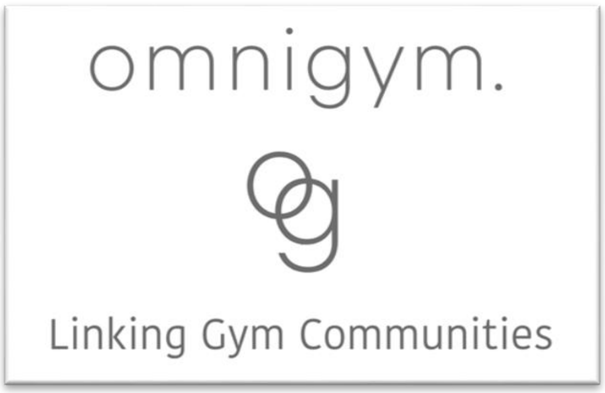
2.14 UML Diagram

To support the continued development of the Omnigym Social App, a UML diagram was created and updated to reflect the second iteration of the project to reflect the system's evolving architecture and features. This diagram helps visualize how different components of the application interact, clarifies responsibility among user roles, and guides both backend logic and frontend design. By maintaining and refining our UML model, the team can uphold consistency between planning, implementation, and future scalability. The updated diagram can be found in Appendix B of this documentation.

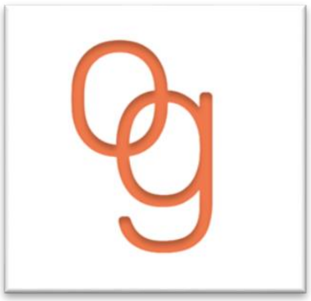
2.15 Brandy Identity

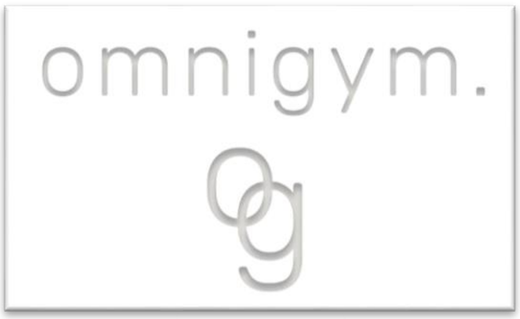
1. Logo Designs

The first design was originally created on the Procreate drawing app by Violet. Due to font constraints in Figma, the logo design changed to the bottom two.

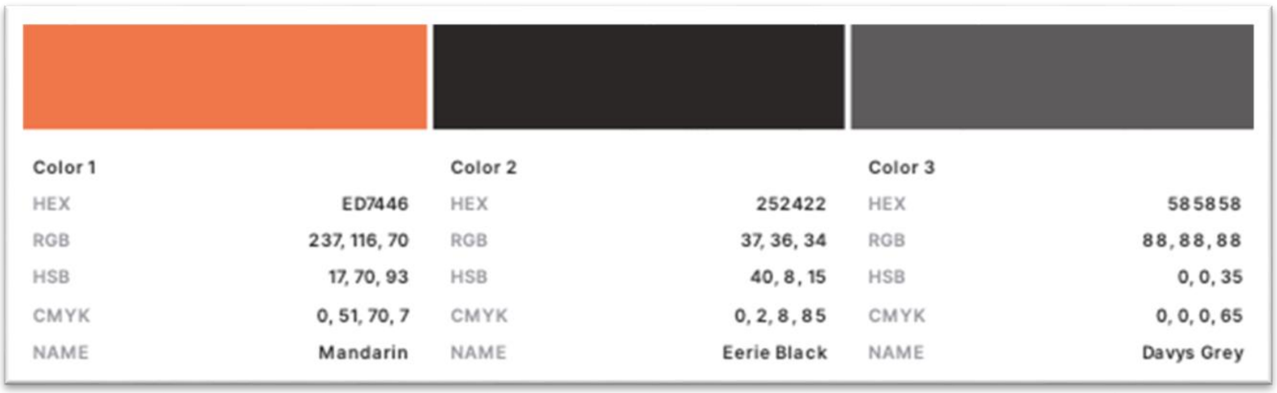


Logo Font: Rubik, light

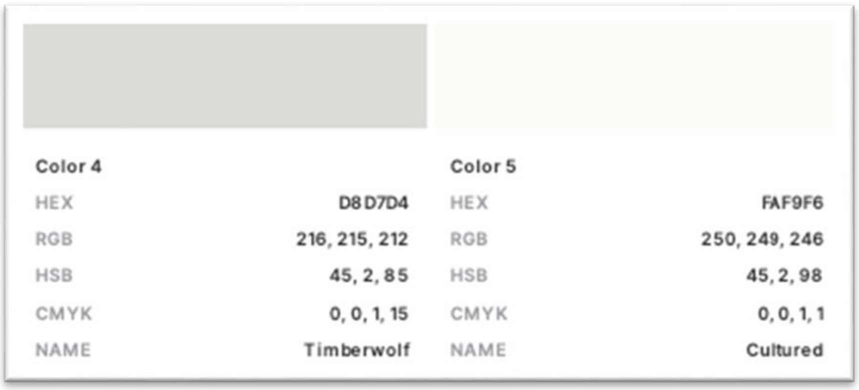




2. Color Palette



Color 1		Color 2		Color 3	
HEX	ED7446	HEX	252422	HEX	585858
RGB	237, 116, 70	RGB	37, 36, 34	RGB	88, 88, 88
HSB	17, 70, 93	HSB	40, 8, 15	HSB	0, 0, 35
CMYK	0, 51, 70, 7	CMYK	0, 2, 8, 85	CMYK	0, 0, 0, 65
NAME	Mandarin	NAME	Eerie Black	NAME	Davys Grey



Color 4		Color 5	
HEX	D8D7D4	HEX	FAF9F6
RGB	216, 215, 212	RGB	250, 249, 246
HSB	45, 2, 85	HSB	45, 2, 98
CMYK	0, 0, 1, 15	CMYK	0, 0, 1, 1
NAME	Timberwolf	NAME	Cultured

3. Use Cases & User Stories

The following use cases and user stories reflect core features prioritized in Iteration 2 of the Omnigym Social App. These user interactions are designed to promote engagement, social connection, and functional value for gym members and staff. Each use case captures the expected flow of system behavior from the perspective of the user.

3.1 Use Cases

Use Case 1: View Leaderboard

Actor: Gym Member

Goal: View personal ranking among members at their gym

Precondition: User is logged in and belongs to a verified gym

Main Flow:

1. User logs into the app
2. User navigates to the "Leaderboard" tab
3. System displays the leaderboard for the user's affiliated gym

Use Case 2: Submit Personal Record (PR)

Actor: Gym Member

Goal: Submit a personal best for a specific lift (e.g., squat, bench press)

Precondition: User is authenticated

Main Flow:

1. User navigates to the "Submit PR" section
2. User selects the lift type and enters weight and reps
3. System saves the entry and updates the gym leaderboard accordingly

Use Case 3: Participate in Gym Event

Actor: Gym Member

Goal: Sign up for an upcoming gym-hosted event (e.g., competition, meet-up)

Precondition: The event is already published by the gym admin

Main Flow:

1. User browses the “Events” tab
2. User selects an upcoming event
3. User clicks “Join Event”
4. System registers the user and confirms participation

Use Case 4: Post to Social Feed

Actor: Gym Member

Goal: Share workout updates or photos with other gym members

Precondition: User has a verified gym membership

Main Flow:

1. User opens the “Social Feed” tab
2. User creates a post (text, photo, or both)
3. System adds the post to the feed visible to other members at the same gym

Use Case 5: Create Gym Event

Actor: Gym Admin

Goal: Create and publish a new event for gym members

Precondition: Admin is logged in with elevated permissions

Main Flow:

1. Admin navigates to the “Create Event” section
2. Admin inputs event details (title, description, date, location)

3. System publishes the event and notifies relevant gym members

3.2 User Stories

- **User Story 1:**

  “As a gym admin, I want to create and manage gym events so that I can promote member participation and foster a stronger gym community.”

- **User Story 2:**

  “As a gym member, I want to share my workout updates and photos so that I can connect and engage with other members at my gym.”

4. User Guide

4.1 Installation Instructions

This guide will help users download the project, install dependencies, and run the app using React Native + Expo. Users can compile and view the app on web, Android, and iOS simulators while it is still being developed using Expo Go. The instructions are below and in the ReadMe section of the GitHub Repo.

1. Download and Install Node.js

https://nodejs.org/en

In terminal:

Check if it was installed properly or if you already have it:

```
node -v
```

2. Install Homebrew package

MacOS:

1. In terminal (Mac):

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

- *Follow on-screen instructions: input password + click enter*

2. Once installation completes, add homebrew to be an accessible path:

```
echo 'eval "$(/opt/homebrew/bin/brew shellenv)"' >> ~/.zshrc

eval "$(/opt/homebrew/bin/brew shellenv)"
```

- *If using **Bash**, replace ~/.zshrc with ~/.bash_profile.*

3. Verify installation was successful

```
brew --version
```

- *Output should be similar to: Homebrew 4.x.x*

Windows OS:

1. In terminal (Windows —> Using Windows Subsystem for Linux (WSL) or Git Bash):

   ** To install WSL (recommended instead of Git Bash):*

```
       wsl --install
```

- *After installing, download Ubuntu. reboot computer.*

If you don't have Ubuntu installed, run this command instead:

```
       wsl –install -d Ubuntu
```

- If using WSL (recommended), open Ubuntu. If any issues arise, preview download instructions from their website to debug.
- If newly downloaded, create account and re-enter password for verification.

Next, run this command in Ubuntu and follow prompts:

```
 /bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

For WSL Users, do not proceed with Git Bash steps below.

- Go to step 2 instructions further below after entering above command.
- Remaining commands in document will be done in Ubuntu.

1. If using Git Bash instead of WSL:

   ***Git Bash requires additional dependencies not listed*

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

   - *Follow on-screen instructions: input password + click enter*

2. Once installation completes on Git Bash, add homebrew to be an accessible path:

```
echo 'eval "$(/home/linuxbrew/.linuxbrew/bin/brew shellenv)"' >>
~/.bashrc

eval "$(/home/linuxbrew/.linuxbrew/bin/brew shellenv)"
```

3. Verify installation was successful

```
brew --version
```

   - *Output should be similar to: Homebrew 4.x.x*

3. Running Expo with React Native (Frontend)

- https://docs.expo.dev/get-started/set-up-your-environment/?platform=android&device=simulated

Tools for Development Guidance (for development only):

- https://docs.expo.dev/develop/tools/

Recommendations:

- Download Expo Tools extension in VSCode
- Download React Native Tools extension in VSCode

In Terminal:

1. First Time Downloading:

```
 cd FullStack/Frontend

 npm list -g expo-cli

npm install

brew install node

brew install watchman
```

- Watchman additional info:

  https://facebook.github.io/watchman/docs/install#macos

2. Open iOS and/or Android Studio simulators before running projects.

3. Update these commands:

```
npm install @react-native-community/datetimepicker@8.2.0 @react-
native-picker/picker@2.9.0 expo-constants@~17.0.8 expo-router@~4.0.19
expo@~52.0.38 jest-expo@~52.0.6

npm audit fix

npm install react-native-select-dropdown

npx expo install expo-file-system

npx expo install @supabase/supabase-js @react-native-async-
storage/async-storage

npm install react-native-url-polyfill
```

Running project:

Ensure you're already in "FullStack/Frontend" directory.

1. Run project

```
npx expo start
```

2. Run project and clear cache

```
npx expo start --clear
```

3. After running:

- You can run both simulators at the same time as long as these commands are performed.

Open project in iOS simulator (must have Mac and Xcode):

```
i
```

Open project in Android Studio simulator (must have downloaded):

```
a
```

Exit running terminal project:

```
cntrl + c
```

4. Setting Up Backend

Supabase documents: https://supabase.com/docs/guides/database/arrays

1. Django Connection:

```
cd omnigymsocialapp/Fullstack/Backend

python -m venv venv
```

MacOS:

```
source venv/bin/activate
```

Windows:

```
venv\Scripts\activate
```

```
pip install django djangorestframework

django-admin startproject server .

python manage.py runserver

pip install djangorestframework-simplejwt

pip install python-dotenv

pip install psycopg2-binary
```

2. Create and Apply DB:

```
python manage.py makemigrations

python manage.py migrate
```

3. Run:

```
python manage.py runserver
```

4. Super user:

```
python manage.py createsuperuser

python manage.py runserver
```

5. Connect Supabase to Django:

```
pip install psycopg2 python-dotenv

python test_connection.py

pip install PyJWT requests

pip install dj-database-url
```

6. Troubleshooting:

- Ensure your Node.js and npm versions meet the minimum requirements.
- Confirm that Android Studio (and the Android SDK) and Xcode (Mac only) are correctly installed and configured.
- Make sure the Expo CLI is installed globally and you're using the latest version.
- For further assistance, refer to the React Native Documentation and the Expo Documentation.

4.2 Licensing

Omnigym Social App is provided under a proprietary software license designed specifically for gym members. The key licensing details include:

- Authorized Use:

The app is licensed for personal use only by individuals with an active, verified membership at Omnigym-affiliated gyms.

- Usage Limitations:
  - Redistribution, resale, or sublicensing of the app (or any derivative content) is strictly prohibited.
  - Reverse-engineering, decompiling, or otherwise modifying the app's software is not permitted without explicit written consent from the owners.
- Open-Source Components:

  Omnigym social app incorporates various open-source libraries. Users must comply with the terms of these open-source licenses as indicated in their project's license file.

4.3 Operator and Subcontractor Indemnity

To protect the developers and associated parties, the following indemnity provisions apply:

Indemnification Clause:

Users agree to indemnify and hold harmless the developers, operators, and any subcontractors involved in Omnigym Social App from any claims, damages, or liabilities that arise from:

- Unauthorized use or modifications of the app.
- Misuse of the software leading to data breaches or security incidents.
- Any third-party claims related to content posted or modifications made by the user.

Liability Limitations:

The developers and operators are not liable for damages or losses arising from:

- Security breaches, including those resulting from user error or external cyberattacks.
- Third-party modifications or integrations that deviate from the approved project configuration.

4.4 Summary of User Agreement

The Omnigym social app User Agreement sets forth the terms and conditions for using the app, which are summarized below:

- Membership and Access:
    - Only users with a verified membership from an Omnigym-affiliated gym may register and use the app.
    - During registration, details such as gym name, location, and membership ID are validated against the gym's database.
    - Inactive memberships result in login failures, with users directed to submit a ticket for verification or membership changes.
- User Responsibilities and Content:
    - Users must provide accurate, complete information during registration.
    - All content (text, images, videos) submitted by users must adhere to applicable laws and the app's content policies.
    - By posting content, users grant Omnigym social app a non-exclusive, worldwide, royalty-free license to use, modify, and distribute the submitted material for app-related purposes.
- App Features and Data Protection:
    - The app facilitates community-building through features like leaderboards, private messaging, workout plan creation, and profile management.
    - User data is collected and stored securely, including personal details and fitness metrics, and is used solely for enhancing the user experience.
    - Users have control over their profile visibility and the information displayed (e.g., fitness progress, PR songs).

- Compliance and Updates:
    - The agreement covers compliance with relevant data protection laws (e.g., GDPR, CCPA) and industry-specific standards.
    - Any changes to the terms or the app’s functionality will be communicated to users, with continued use implying acceptance of updated terms.

5. Stakeholder Requirements

We identified stakeholder requirements by interviewing gym members and peers, sharing personal experiences, analyzing the causes of gym-timidation and how Omnigym can address these issues for primary users while increasing gym participation and sales, and conducting prototype design feedback sessions. This process allowed us to understand each group's unique needs and challenges—for instance, gym owners desired ways to boost engagement and sales, while community members sought a friendly, supportive environment to overcome gym-timidation and increase motivation. Personal trainers and staff members highlighted the difficulty of approaching members through cold calls and expressed a need for a more personable method of breaking the ice by leveraging prior knowledge of the members. Investors emphasized the importance of a solid, growth-oriented revenue model, and our app support team stressed the need for a robust, secure backend that supports continuous updates along with organized documentation and version control. The requirements matrix for this section can be found in section 2.6 of this documentation.

Our final vision for Omnigym addresses these diverse requirements by integrating tailored features for each stakeholder group. For gym owners, the platform offers comprehensive analytics and engagement tools designed to increase revenue and member loyalty. Primary users benefit from community-centric features such as messaging, leaderboards, workout tracking, and social interactions that directly counteract gym-timidation. Personal trainers have access to member profiles to review fitness goals and metrics, as well as a dedicated communication channel to connect with clients, while gym employees enjoy an intuitive interface for managing daily operations and event postings. Investors receive a clear growth strategy with opportunities for promotional expansion, and our app support team is empowered by a streamlined infrastructure that facilitates prompt updates and high-quality customer service. Overall, our iterative, feedback-driven approach has allowed us to create a comprehensive solution that meets—and often exceeds—the expectations of all stakeholders.

6. Final Vision Alignment

Our final vision ensured that Omnigym met the needs of users while maintaining regulatory compliance and a user-friendly experience.

6.1 Feature Completeness

Omnigym Social App includes all essential features such as secure authentication, workout tracking, scheduling, and social networking, ensuring a fully functional and seamless experience for gym members and trainers. Continuous updates and feature expansions are planned based on user feedback.

6.2 Regulatory Compliance

The platform adheres to data privacy regulations such as GDPR and CCPA, implementing encryption protocols and secure authentication methods. Compliance with fitness industry standards ensures that both user data and operational processes meet legal and ethical requirements.

6.3 User-Centric Design

UI/UX decisions were based on user research and verbal surveying, feedback loops, and iterative design improvements. The intuitive layout, accessibility considerations, and smooth navigation contribute to an engaging and efficient user experience.

7. Software Requirement Specification

The Software Requirements Specification (SRS) document serves as a contract between the development team and the stakeholders, defining the functional and non-functional requirements necessary for the successful implementation of the Omnigym Social App. It establishes a firm foundation for system design, supports validation and verification, and ensures clarity in communication between developers, customers, and project managers.

7.1 Purpose of SRS

The purpose of this SRS document is to provide a clear understanding of the system requirements for stakeholders. It will act as a reference document throughout the project lifecycle to ensure we are on track and meeting SRS goals. These specifications will also define the scope, performance expectations, and constraints of the Omnigym Social App. Overall, the SRS will serve as our contract between the development team and stakeholders to resolve any disputes regarding functionality.

7.2 Scope

The Omnigym Social App is a cross-platform mobile application developed using React Native with Expo to support both iOS and Android platforms. It is designed for verified members of affiliated gyms to create accounts, track workouts, submit personal records, participate in gym-hosted events, and engage socially through community features. The backend is implemented using Django REST Framework and PostgreSQL, integrated via Supabase for authentication and real-time data synchronization.

Key Objectives:

- Support multiple user roles (gym members, trainers, admins) simultaneously.

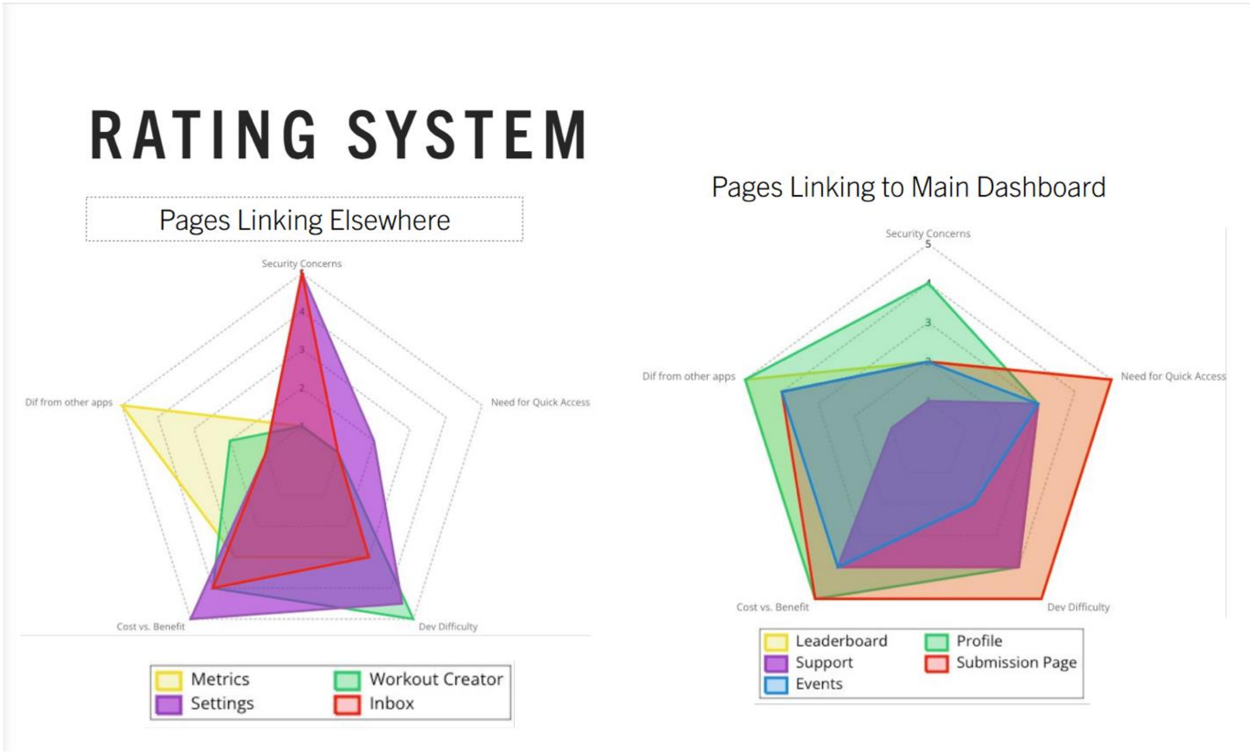
- Allow secure authentication and role-based access control like updating databases from admins or updating profiles and form submissions for users.
- Enable workout routine creation using an AI assisted generator, workout and metrics tracking, leaderboard competitions, event postings, profile customizability, and social engagement through messaging and leaderboard posts.
- Real-time updates and interaction by users and admins.
- Ensure high availability, performance, and data security.
- Comply with industry regulations regarding data protection and privacy.

7.3 Functional Requirements

The system's functional requirements define what the software must do:

- **User Registration and Authentication:** Secure sign-up and login functionality with JWT-based authentication through Supabase. Registration must be confirmed using backend database connections to verify gym memberships. When tables are updated by the admin, user memberships must be re-verified in the system. Any users who are no longer active must receive a warning that their profile will be discontinued until membership is either updated or shifted to a different affiliated gym.
- **Workout and Progress Tracking:** Users can log their workouts, track fitness metrics, update and display personal records.
- **Personal Record (PR) Submissions:** Allows gym members to submit PR forms for leaderboards.
- **Event Participation:** Users can browse and join events published by the gym admin.
- **Social Features** (Planned)**:** Enables users to interact with trainers and other members through leaderboard competitions, profiles, and inbox messaging.

- **Customizability:** Allow users to edit profile features and descriptions, set profile to public/private, change type of measurement units used, and check Wilks 2 score.



7.4 Non-Functional Requirements

These requirements define the system's quality attributes:

- **Performance:** The system must support multiple users logged-in and their records simultaneously without performance degradation.
- **Security:** All user data must be encrypted, and authentication must follow OAuth2 and JWT standards. Hashing should be done using the Supabase built-in *User* table. Database calls to Supabase should use Django and PostgreSQL for best security practices.
- **Scalability:** The backend must support future expansion with minimal refactoring utilizing SOLID concepts and encapsulation.

- **Usability:** The user interface should be intuitive, mobile-friendly, and accessible to users with different fitness goals and varying technical backgrounds. Offline access to cached workout routines is planned for the next iteration.

7.5 External Interface Requirements

- **User Interfaces:** Use React Native with Expo to optimize development for both iOS and Android mobile platforms.
- **Hardware Interfaces:** Must be compatible with Android and iOS devices and web browsers for editing.
- **Software Interfaces:** Integrates with Supabase for real-time data synchronization and Django backend APIs.
- **Communication Interfaces:** Uses RESTful APIs for seamless data exchange and calls.

7.6 Design Constraints

- Our designs need to follow the Agile development methodologies.
- Must comply with GDPR and CCPA regulations regarding user data privacy.
- The system must be mobile friendly with Android and iOS devices.

7.7 Performance and Security Specifications

- Response Time: All user interactions must be processed within 2 seconds under normal load.
- Availability: The system must maintain 99.9% uptime and use a caching system for saved routines. Cached routines should be accessible even if offline.
- Data Protection: The app must implement end-to-end encryption and secure authentication mechanisms to keep users safe and stakeholders satisfied with performance and safety measures that were taken.

7.8 Verification and Validation

- Unit Testing: Each module must undergo automated testing to ensure correctness. We will use both online simulators and connected iOS and Android devices to test compatibility, design appearance and app performance.
- Integration Testing: API endpoints and data flow must be tested using Postman and CI/CD pipelines.
- User Acceptance Testing (UAT): Stakeholder feedback must be incorporated before final deployment.

8. Meeting Stakeholder Needs

To establish customer needs, we conducted a survey of approximately two hundred viewers, which helped validate the app name and gather initial feedback. Some viewers responded with additional feedback revealing that they were excited about a digital social platform where they can track their workouts—eliminating the need for clipboards—and engage in friendly competitions with fellow members. Some expressed concerns about feeling intimidated at the gym due to uncertainty about what to do, a challenge our workout routine creator aims to address by offering an easy way to create and track workout routines.

For the other stakeholders, gym owners, personal trainers, gym employees, investors, and our app support team—we relied on personal experience and peer discussions to understand their needs. Gym owners look for tools that boost engagement and drive sales, while personal trainers and staff members highlighted the importance of efficient communication and client management. Investors prioritized a robust, growth-oriented revenue model, and our support team emphasized the need for a secure, easily updatable backend with organized documentation and version control.

Although this initial iteration is based on preliminary feedback and internal insights, we plan to conduct further user testing in iteration two to refine the app's functionality and interface based on direct user input. Overall, our iterative, feedback-driven approach has enabled us to create a solution that addresses the diverse requirements of all stakeholders while delivering an engaging experience for gym members.

9. Team Collaboration and Performance

9.1 Common Goal Overview

Our team is developing a gym social app that is exclusively available to affiliated gyms, like Handshake's affiliation with colleges. The app focuses on providing a platform for creating gym routines, convenient fitness tracking, and enhanced gym member engagement.

9.2 Team Collaboration and Effectiveness

- Our team effectively collaborated by:
- Assigning clear roles and responsibilities: Each team member had a defined role, ensuring accountability and an efficient workflow.
- Maintaining open communication: We held weekly meetings, conducted regular reviews, and utilized Microsoft Teams to stay aligned.
- Using collaborative tools: We leveraged Microsoft Teams, GitHub, Excel Gantt Charts, MS Planner, Editor, and Figma to streamline our workflow and project management.

9.3 Team Members and Roles

- **Violet** – Executive Lead, Operations Manager, Full Stack Engineer, Graphic Designer
- **Jawad** – Project Manager, Tech Lead, Android Lead
- **Maruf** – iOS and Frontend Engineer
- **Kliman** – Scrum Master, Backend Developer, Database Support
- **Robert** – Frontend Engineer, Middleware Support
- **Viktor** – Frontend Support, Backend Support

9.4 Tech Stack

- **Frontend:** React Native + Expo

- **Backend:** Django
- **Database & Cloud Services:** Supabase/PostgreSQL
- **Tools:** Microsoft Teams, GitHub, Excel Gantt Charts, MS Planner, Figma

9.5 Ability to Function Effectively

As the Tech Group Lead, I played a crucial role in overseeing the technical direction of our Fitness Leaderboard App. I was responsible for Android development, SQL database management, and backend implementation, while also ensuring that the chosen tech stack (React Native with Expo, Django, and Supabase) aligned with our project goals. Beyond development, I took on project management responsibilities, creating Gantt charts for goal tracking, conducting weekly meetings on Microsoft Teams, and managing MS Planner boards to assign key tasks. To build an efficient team, I took the time to learn about each member's skills and interests to ensure they were assigned roles where they could contribute effectively. I actively monitored team progress, provided technical guidance, and 0072esearched solutions to overcome challenges efficiently. When conflicts arose, I offered constructive feedback and implemented strategies to keep the team aligned. My ability to balance technical execution, leadership, and team coordination ensured smooth collaboration and effective progress toward our common goal.
Our structured approach, combined with strong collaboration and the right tools, has helped us efficiently develop and manage our Fitness Leaderboard App.

10. Project Criteria Satisfaction

The Omnigym app successfully satisfies all planned features for Iteration 1 and has made significant progress toward implementing several Iteration 2 features. Users can register, log in, and access the home page, which includes navigable sections such as profile, support, events, and settings. As of this iteration, development is underway for workout tracking, AI-assisted routine generation, real-time messaging, and social networking — all of which aim to increase engagement, foster motivation, and build a stronger sense of community among gym members.

The app is being developed using the Agile-Scrum methodology, which allows for flexibility, iterative delivery, and rapid response to feedback. The tech stack was refined between iterations to improve performance and scalability. While the early prototype used Ionic React with Python, SQLite3, and Firebase, the current architecture has been upgraded to use React Native with Expo for cross-platform mobile development and a Django + PostgreSQL backend, hosted via Supabase. This allows us to work from a unified codebase that supports both iOS and Android, streamlining development and deployment.

We've applied SOLID principles and best practices to maintain a clean and modular architecture. Our codebase features meaningful identifiers, descriptive filenames, and organized folder structures. Adherence to PEP8 styling, modular components, and proper indentation ensures clarity and maintainability. The backend API is tested using Postman, and unit testing was introduced for core features like authentication, workout tracking, and leaderboard updates.

Version control is managed through GitHub, and all commits are reviewed before merging. The team follows a consistent Git workflow using feature branches and pull requests. Code reviews are conducted regularly, improving quality and encouraging team knowledge-sharing. Issues and bugs are tracked using GitHub Issues and resolved based on priority. A major hurdle encountered during Iteration 1 involved Android Gradle build errors, which led to our decision to transition to React Native and Expo to simplify deployment across platforms (GitHub).

To support future developers and users, a comprehensive README file was created, detailing the project's purpose, core features, contributors, and an installation guide outlining system requirements and setup steps.

From a project management perspective, sprint planning was conducted using Microsoft Planner, and meeting minutes were documented in Microsoft Teams. Each sprint began with clearly defined goals and task assignments based on team expertise. Progress was tracked regularly, and retrospectives were held at the end of each sprint to assess outcomes and identify areas for improvement. These reflections were documented and used to adapt processes in future sprints.

Collaboration remained a key strength throughout development. We utilized Microsoft Teams, GitHub, SharePoint, and Figma to communicate effectively, share assets, and keep development aligned. Daily stand-ups helped surface issues early, and asynchronous check-ins ensured flexibility. Our team even created tutorial videos to support one another with tool usage, such as GitHub version control and merging best practices (GitHub).

In summary, the Omnigym app is meeting its original goals while evolving based on user needs and technical feasibility. Iteration 2 reflects a shift toward scalability, maintainability, and user-centered design, setting the stage for even more impactful features in upcoming development cycles.

11. Stability Assessment

REMEMBER TO SHOW YOUR WORK SOMEHOW (if it's based on nodes, charts, etc, then discuss them or show them and the numbers/values you used)

11.1 Customization Index

11.2 Structural Complexity

11.3 System Complexity

11.3 Morphology Metrics

11.4 Interface Metrics

11.5 Aesthetics Metrics

11.6 Content Metrics

11.7 Source Code Metrics (Halstead)

11.8 Goal-Driven Metrics

12. References

GitHub. (n.d.). *About repositories*. GitHub Docs: Repositories. https://docs.github.com/en/repositories/creating-and-managing-repositories/about-repositories

Meta Platforms, Inc. (2025, April 14). Introduction · REACT NATIVE. React Native RSS. https://reactnative.dev/docs/getting-started

Supabase. (2025, April 4). *Getting started: Supabase docs*. Supabase Docs: Getting Started. https://supabase.com/docs/guides/getting-started

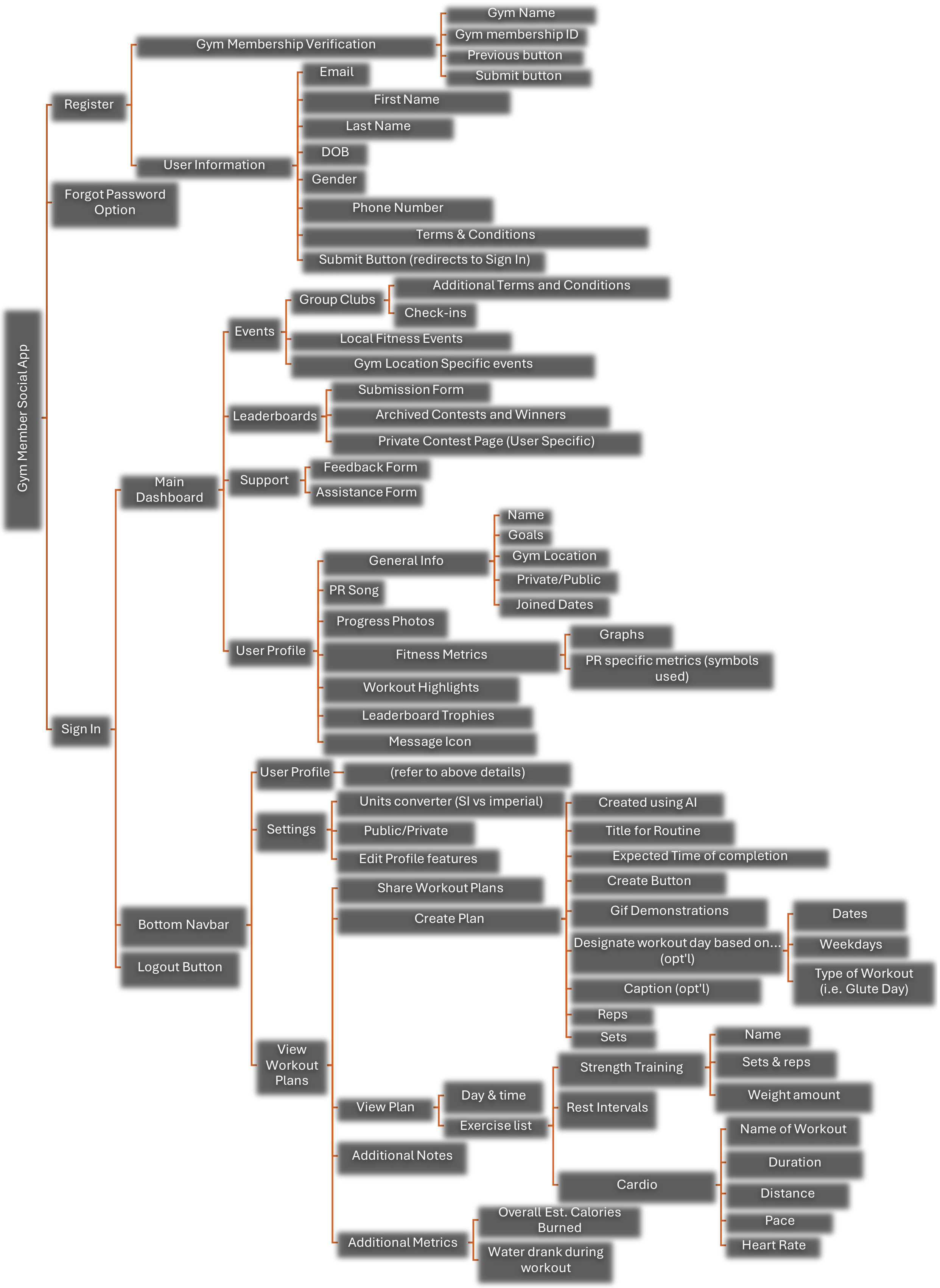
W3Schools Online Web Tutorials. (n.d.). *W3schools.com*. W3Schools: PostgreSQL Tutorial. https://www.w3schools.com/postgresql/

TODO (citations to add):

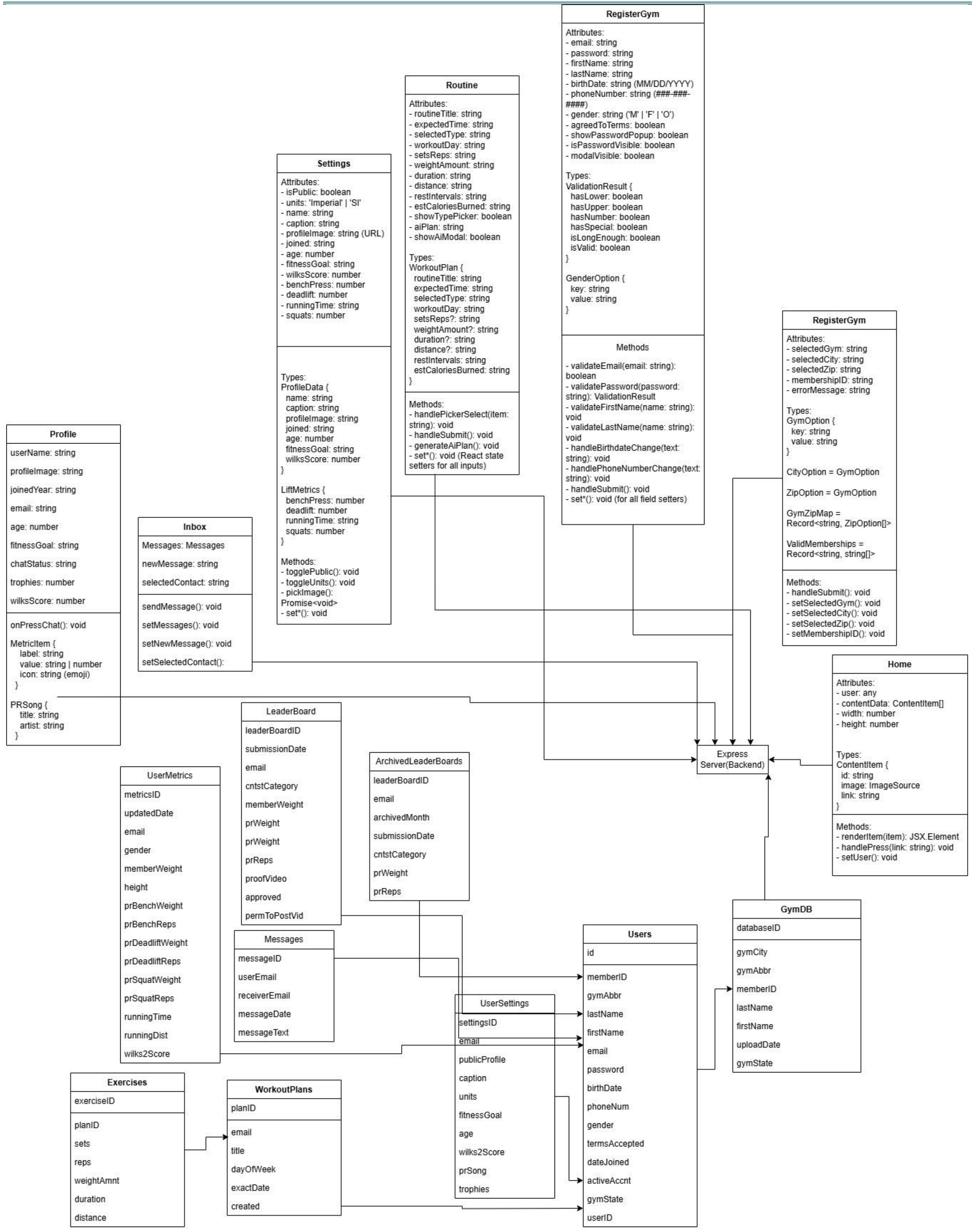
https://ionic.io/ionicons

13. Appendix A

This appendix shows the feature creation outline for Omnigym during our initial team meeting that we originally created on Edotor and further expanded on. Due to its close attention to feature details being included, the best viewability is virtual and zooming into the page. Graph can be viewed on the next page.


Gym Member Social App
Register
Gym Membership Verification
Gym Name
Gym membership ID
Previous button
Submit button
User Information
Email
First Name
Last Name
DOB
Gender
Phone Number
Terms & Conditions
Submit Button (redirects to Sign In)
Forgot Password Option
Sign In
Main Dashboard
Events
Group Clubs
Additional Terms and Conditions
Check-ins
Local Fitness Events
Gym Location Specific events
Leaderboards
Submission Form
Archived Contests and Winners
Private Contest Page (User Specific)
Support
Feedback Form
Assistance Form
User Profile
General Info
Name
Goals
Gym Location
Private/Public
Joined Dates
PR Song
Progress Photos
Fitness Metrics
Graphs
PR specific metrics (symbols used)
Workout Highlights
Leaderboard Trophies
Message Icon
Bottom Navbar
User Profile
(refer to above details)
Settings
Units converter (SI vs imperial)
Public/Private
Edit Profile features
View Workout Plans
Share Workout Plans
Create Plan
Created using AI
Title for Routine
Expected Time of completion
Create Button
Gif Demonstrations
Designate workout day based on... (opt'l)
Dates
Weekdays
Type of Workout (i.e. Glute Day)
Caption (opt'l)
Reps
Sets
View Plan
Day & time
Exercise list
Strength Training
Name
Sets & reps
Weight amount
Rest Intervals
Cardio
Name of Workout
Duration
Distance
Pace
Heart Rate
Additional Notes
Additional Metrics
Overall Est. Calories Burned
Water drank during workout
Logout Button

14. Appendix B



15. Appendix C

The following pages are imported records of sprint meetings the team tracked with the projects progress information and overall details of meetings and tasks that were discussed and/or completed so far.

Week 1

Client:	---

Software Engineers:	The Dream Team (Group 13)	**Scrum Master:**	Jawad Rashid

Sprint:	Date:	01/16/2025	Time:	2 hrs 30 min	
Attendees:	Kliman Darawish		Abdulla Maruf		
	Robert Simovski		Jawad Rashid		
	Viktor Gjorgjevski		Violet Yousif		
Achievements:	Completed Edotor Graph. Decided on app concept.				

Product Backlog: (In-progress tasks and by whom, forms to be created)
Group was tasked with forming and deciding on an app concept.
Create a PowerPoint with our process and decisions.

Sprint Goals: (New requirements/events, existing goals)
Decide on app concept using Edotor graph site and rating system.
Decide what is important to include in rating system.

Team Availability: (Tasks and research given to members on individual time)
Group – Research app name concepts. Check trademarking and app store availability.
Jawad – Add Project Management schedule. Research project tech stack.
Violet – Create Logo concept.
Kliman – Research AI workout scheduler.
Maruf – Research data storage options.

Assign Backlog Items: (Assigned tasks pertaining to completion of Backlog goal)
Group – Rate main dashboard pages using rating system.
Group – Research ideas pertaining to main dashboard linked pages and features like linking AI capabilities, analytics, profiles, etc.
Violet – Will create PowerPoint presentation and Scrum Meeting form.

Retrospective: (What went right? What went wrong?)
App concept was decided on. Team did great thinking outside the box for feature concepts.
A workplan with deadlines and dividing project into parts could improve flow. Relocating during in-person labs may reduce distractions and improve productivity and focus.

Week 4

Client:	Forestview		
Software Engineers:	The Dream Team (Group 13)	**Scrum Master:**	Kliman Darawish

Sprint:	Date:	02/06/2025	Time:	5 hrs
Last Sprint:	Date:	01/30/2025	Time:	4 hrs 30 min
Attendees:	Kliman Darawish		Abdulla Maruf	
	Robert Simovski		Jawad Rashid	
	Viktor Gjorgjevski		Violet Yousif	
Achievements:	Created a mockup of the Starburst points.			

Product Backlog: (In-progress tasks and by whom, forms to be created)
Group – Work on starburst details
Discuss framework and programming findings and research.

Sprint Goals: (New requirements/events, existing goals)
Finish Starburst discussion.
Finish presentation to address starburst concerns.
Begin discussion on getting database and programs started.

Team Availability: (Tasks and research given to members on individual time)
Group – Add personal notes to designated slides.
Viktor and Robert – Create Presentation
Violet – Continue with mockup designs. Create a mockup of database table info.

Assign Backlog Items: (Assigned tasks pertaining to completion of Backlog goal)
Group – Start reviewing and researching iteration project requirements.

Retrospective: (What went right? What went wrong?)
Team engaged well with each other and members were comfortable with whatever slides they were assigned.
Still need to delegate tasks per member more evenly.

Week 5

Client:	Forestview		
Software Engineers:	The Dream Team (Group 13)	**Scrum Master:**	Kliman Darawish

Sprint:	Date:	02/13/2025	Time:	6 hrs
Last Sprint:	Date:	02/06/2025	Time:	5 hrs
Attendees:	Kliman Darawish		Abdulla Maruf	
	Robert Simovski		Jawad Rashid	
	Viktor Gjorgjevski		Violet Yousif	
Achievements:	Decided on team roles. Begin programming.			

Product Backlog: (In-progress tasks and by whom, forms to be created)
Group - Split group up into Android and Apple teams
Discuss frameworks for frontend and backend

Sprint Goals: (New requirements/events, existing goals)
Work on project one presentation
Commit to Frameworks and languages.

Team Availability: (Tasks and research given to members on individual time)
Group - Learn how to use frameworks
Robert and Violet - Create frontend programs
Jawad - Create PowerPoint and update slides
Adbulla - Create firebase tables
Viktor - Research Django and assist Kliman
Kliman - Work on backend development/research

Assign Backlog Items: (Assigned tasks pertaining to completion of Backlog goal)
Violet - team leader, Robert and Abdulla assisting (primarily frontend and IOS support)
Jawad - team leader, Viktor and Kliman assisting (primarily backend and Android support)

Retrospective: (What went right? What went wrong?)
Team did well delegating roles, cleared up assignment details.
Choosing what framework to use was a bit confusing at first.

Week 6

Client:	Forestview		
Software Engineers:	The Dream Team (Group 13)	**Scrum Master:**	Kliman Darawish

Sprint:	Date:	02/20/2025	Time:	8 hrs
Last Sprint:	Date:	02/13/2025	Time:	6 hrs
Attendees:	Kliman Darawish		Abdulla Maruf	
	Robert Simovski		Jawad Rashid	
	Viktor Gjorgjevski		Violet Yousif	
Achievements:	Completed Iteration 1			

Product Backlog: (In-progress tasks and by whom, forms to be created)
Group – Finish Iteration 1 program, documentation, and Presentation.
Jawad, Viktor, Kliman – Research Django

Sprint Goals: (New requirements/events, existing goals)
Group – Discuss what needs to be completed still.
Commit to new Frameworks and languages.

Team Availability: (Tasks and research given to members on individual time)
Violet and Jawad – Delegate team for next iteration
Robert – Improve/work on connecting HTML code to frontend app
Adbulla – Update database headers
Kliman and Viktor – Begin converting backend code for React.

Assign Backlog Items: (Assigned tasks pertaining to completion of Backlog goal)
Group – Finish documentation. Requirement tasks not assigned because it will be based on comfortability of answering questions per person.

Retrospective: (What went right? What went wrong?)
Presentation went well. Team leaders did well delegating. Audience was engaged and enthusiastic about the app concept. Received valuable feedback from the audience and Forestview CEO.
Establishing meeting times seems to be difficult with varying schedules. CEO recommended being more "hype" when presenting but liked that it came out when engaging with audience.

WEEK 7

Client:	Forestview

Product Owners:	The Dream Team (Group 13)	**Scrum Master:**	Kliman Darawish

Sprint:	Date:	02/27/2025	Time:	6 hrs
Last Sprint:	Date:	02/20/2025	Time:	8 hrs
Attendees:	Kliman Darawish		Abdulla Maruf	
	Robert Simovski		Jawad Rashid	
	Viktor Gjorgjevski		Violet Yousif	
Achievements:	Decided to switch frontend to React Native + Expo, & Django for backend			

Product Backlog: (In-progress tasks and by whom, forms to be created)
Group - Assign tasks for iteration 2
Jawad, Viktor, Kliman - Implement Django Rest Framework

Sprint Goals: (New requirements/events, existing goals)
Group - Work on individual tasks
Commit to new Fullstack framework

Team Availability: (Tasks and research given to members on individual time)
Violet and Jawad - Delegate team for next iteration
Robert - Convert frontend to React Native
Abdulla - Work on Firebase implementation
Kliman and Viktor - Begin converting backend code from Flask to Django

Assign Backlog Items: (Assigned tasks pertaining to completion of Backlog goal)
Team - Implement Python code to connect to DB and frontend

Retrospective: (What went right? What went wrong?)
Due to restrictions with the previous framework, we had to switch and restart the project.
Incorporating Firebase seems to be more complicated than we thought.

WEEK 8

Client:	Forestview

Product Owners:	The Dream Team (Group 13)	**Scrum Master:**	Kliman Darawish

Sprint:	Date:	03/06/2025	Time:	7 hrs
Last Sprint:	Date:	02/27/2025	Time:	6 hrs
Attendees:	Kliman Darawish		Abdulla Maruf	
	Robert Simovski		Jawad Rashid	
	Viktor Gjorgjevski		Violet Yousif	
Achievements:	Robert and Violet successfully got React Native + Expo to run.			

Product Backlog: (In-progress tasks and by whom, forms to be created)
Jawad & Violet - create SQL database schema
Robert & Violet - Create frontend pages for Android and IOS compatibility

Sprint Goals: (New requirements/events, existing goals)
Research new database that might be more React Native friendly
Update diagrams in iteration 2

Team Availability: (Tasks and research given to members on individual time)
Team - practice git commits to get more comfortable; research tutorials
Robert & Violet - Improve error handling in frontend pages and registration
Kliman & Viktor - test backend framework for errors
Maruf & Jawad - work on database integration

Assign Backlog Items: (Assigned tasks pertaining to completion of Backlog goal)
Incorporate dual authentication for email
Maruf - Look into password hashing with Firebase

Retrospective: (What went right? What went wrong?)
The switch to React Native wasn't too difficult. Backend switch isn't having as smooth of a transition though.

WEEK 9

Client:	Forestview
Product Owners:	The Dream Team (Group 13)
Scrum Master:	Kliman Darawish

Sprint:	Date:	03/13/2025	Time:	7 hrs 30 min
Last Sprint:	Date:	03/06/2025	Time:	7 hrs

Attendees:	Kliman Darawish	Abdulla Maruf
	Robert Simovski	Jawad Rashid
	Viktor Gjorgjevski	Violet Yousif
Achievements:	Frontend Registration is complete.	

Product Backlog: (In-progress tasks and by whom, forms to be created)
Maruf – Investigate authentication
Violet – Update frontend objects to reflect naming in database schema.

Sprint Goals: (New requirements/events, existing goals)
Revise backend code to add better comments for clarity
Discuss Iteration 2 immediate goals and GitHub issues (deleted files reappearing)

Team Availability: (Tasks and research given to members on individual time)
Violet – Build and revise the user installation instructions
Robert – Add new user pages to frontend
Team – Continue working on assigned tasks in MS Planner

Assign Backlog Items: (Assigned tasks pertaining to completion of Backlog goal)
Viktor – Assist Robert & Kliman on frontend/backend
Jawad & Violet – Find alternative to Firebase.

Retrospective: (What went right? What went wrong?)
Frontend registration is almost complete just needs a few tweaks.
Issues with backend and database integration. Still need to connect it to frontend.

WEEK 10

Client:	Forestview		
Product Owners:	The Dream Team (Group 13)	**Scrum Master:**	Kliman Darawish

Sprint:	Date:	03/20/2025	Time:	13 hr 45 min
Last Sprint:	Date:	03/13/2025	Time:	7 hrs 30 min
Attendees:	Kliman Darawish		Abdulla Maruf	
	Robert Simovski		Jawad Rashid	
	Viktor Gjorgjevski		Violet Yousif	
Achievements:	Switching to Supabase for the project database.			

Product Backlog: (In-progress tasks and by whom, forms to be created)
Group - Work on documentation
Violet - Introduce database platform switch (Supabase)

Sprint Goals: (New requirements/events, existing goals)
Preview Supabase with team. Connected everyone's emails.
Switching team roles to compensate for workload and time frame
Violet - Worked with team to get their repos updated and running from GitHub

Team Availability: (Tasks and research given to members on individual time)
Robert & Maruf - Add new frontend pages and debug
Violet & Jawad - Update Supabase tables and research connecting to project
Kliman - Remove deprecated program files and push backend repo (flask and SQLite3)
Viktor - Update UML diagram and other outdated diagrams

Assign Backlog Items: (Assigned tasks pertaining to completion of Backlog goal)
Team - Begin updating documentation from previous iteration requirements
Violet and Jawad - Work on connecting Supabase to updated project. Get User Auth to work in DB.

Retrospective: (What went right? What went wrong?)
Team was excited about the database change.
Behind on progress for backend due to database, need to integrate it fully with frontend
Team was open to team role changes.

WEEK 11

Client:	-- Forestview

Product Owners:	The Dream Team (Group 13)	**Scrum Master:**	Kliman Darawish

Sprint:	Date:	03/27/2000	Time:	12 hr 30 min
Last Sprint:	Date:	03/20/2025	Time:	13 hr 45 min
Attendees:	Kliman Darawish		Abdulla Maruf	
	Robert Simovski		Jawad Rashid	
	Viktor Gjorgjevski		Violet Yousif	
Achievements:	User authentication works. Database integrated with frontend (but broke in merge).			

Product Backlog: (In-progress tasks and by whom, forms to be created)
Jawad & Violet - Completed and updated database tables on Supabase

Sprint Goals: (New requirements/events, existing goals)
Team - Work on comfortability with GitHub and merging repos. Watch short videos Violet made and sent in Teams chat. Also, get repos to run on VSCode.
Discuss urgent matters pertaining to presentation date.

Team Availability: (Tasks and research given to members on individual time)
Violet - Fix backend problem caused by outdated repo's being merged with old dependencies and file structures that broke program links. Assist with full stack issues.
Kliman & Jawad - Work on backend functions in Django for Supabase
Viktor - Create an Admin sign-in profile
Robert & Maruf - Frontend, Integration of full stack
Jawad - Create Presentation. Assist with full stack issues. Update iteration documentation.

Assign Backlog Items: (Assigned tasks pertaining to completion of Backlog goal)
Complete Frontend Pages.
Create Admin pages to allow for direct CSV code parsing into Supabase tables.

Retrospective: (What went right? What went wrong?)
Merge conflicts are breaking main branch code. Pushes with security vulnerabilities are being overlooked due to large dependency files making it difficult to manually fix merge conflicts.
Difficulties connecting the backend to the database and integrating it with the frontend.
Time constraints and overestimating how much we could get complete for this iteration has set us back on progress goals.

WEEK 12

Client:	Forestview
Product Owners:	The Dream Team (Group 13)
Scrum Master:	Kliman Darawish

Sprint:	Date:	04/03/2025	Time:	11 hr 15 min
Last Sprint:	Date:	03/27/2025	Time:	12 hr 30 min
Attendees:	Kliman Darawish		Abdulla Maruf	
	Robert Simovski		Jawad Rashid	
	Viktor Gjorgjevski		Violet Yousif	
Achievements:	Iteration 2 Presentation complete. Frontend User pages mostly complete.			

Product Backlog: (In-progress tasks and by whom, forms to be created)
Viktor – Continue to create Admin Pages for CSV upload and user form submissions
Create Quality Manual and Procedure document per Rubric.

Sprint Goals: (New requirements/events, existing goals)
Assigned tasks per member on Excel doc for Iteration 2 documentation requirements.
Discussed communication strategies and preferences. Reviews are mixed.

Team Availability: (Tasks and research given to members on individual time)
Violet & Jawad – Edit documentation from previous iteration along with new tasks
Maruf – Update host site for iteration 2
Robert – Try to get frontend running again after previous merge conflict
Kliman – Create Quality Manual doc template in folder and share with team.

Assign Backlog Items: (Assigned tasks pertaining to completion of Backlog goal)
Team – work on Iteration 2 and Quality Manual and Procedure documentation.
Team – Continue with expanding vision on project

Retrospective: (What went right? What went wrong?)
Team seemed unaware of the number of new requirements this iteration had until it was addressed in lecture with the introduction of the "Peer Review" document. The time constraint will be an issue with other classes for all of us.
Frontend got most of the pages running for users.
Presentation went well other than minor glitches with Maruf's PowerPoint.