OMNIGYM APP

QUALITY MANUAL & PROCEDURES

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. Vision

1.1 Vision Statement

To redefine the gym experience by building a trusted digital ecosystem where gym-goers, trainers, and administrators thrive together through motivation, connection, and convenient technology.

1.2 Omnigym Slogan

Linking Gym Communities

The digital bridge between gym members, trainers, and their home gyms.

2. Brief Entity Description

2.1 Overview

Application Name: Omnigym

Application Type: Mobile Application / Software Platform

Industry: Health & Fitness Technology

Omnigym is a mobile-based social fitness application designed to enhance gym-goers' workout experience through community-driven engagement and competitions, streamlined gym verification, and personalized tracking features. Built using a full-stack solution (React Native + Django), Omnigym enables verified members of affiliated gyms to connect with fitness peers, track workouts, compete in friendly gym competitions, and stay up to date on local and gym-hosted events — all with the goal of making their gym feel like a second home.

2.2 Core Activities

- **Membership Verification**: Users validate their gym affiliation via integrated gym databases, ensuring a secure and exclusive community.
- **User Registration and Authentication**: A custom two-step user onboarding process guarantees only verified gym members can create accounts and access community features.
- **Social Features**: Planned functionality includes friend connections, gym leaderboards, activity feeds, and motivational tools.
- **Data Management**: The app utilizes a cloud-based PostgreSQL database (via Supabase) for secure and scalable user and gym data handling with password hashing.

- **Platform Scope**: Omnigym is currently in the development phase, with ongoing integration of fixes to routine creation, improving form submissions, and optimizing page layout designs. The application will prioritize security for users and their data. Future enhancements may include reminders, AI routine creator, workout visual aids to demonstrate how to perform a workout, push notifications, and gym analytics for both users and gym administrators.

2.3 Mission Statement

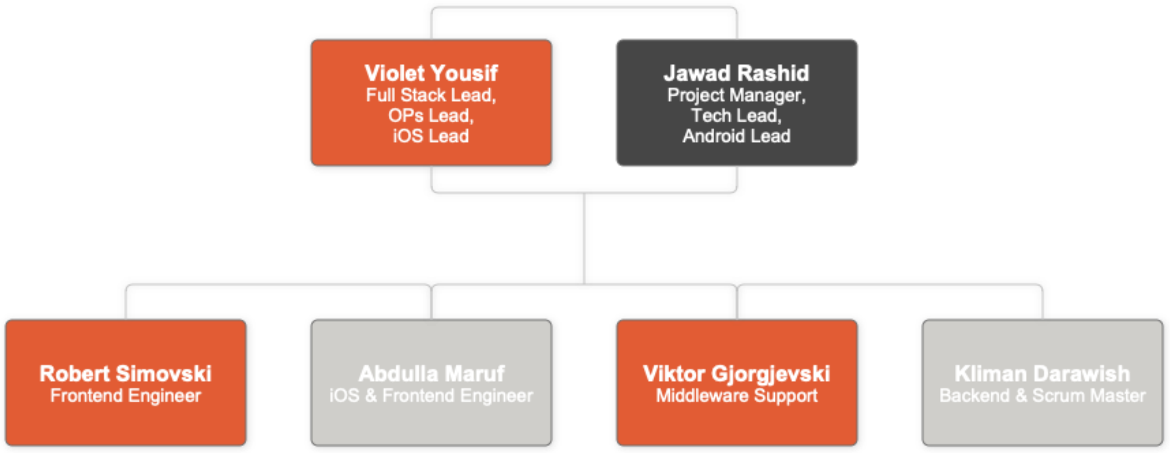
To foster a stronger, more engaging gym community by blending technology, accountability, and social motivation — helping users stay committed to their fitness goals through meaningful digital interactions.

3. Organization Chart

The organizational structure of the Omnigym development team is designed to support a full-stack mobile application workflow. The team is led by two key figures: Violet, who oversees full stack and iOS development, operational procedures, and design implementation; and Jawad, who manages project coordination tasks such as introducing organizational tools, scheduling meetings, and leading Android development as the Tech Lead. Both also contribute to database management and backend implementation.

Under their leadership, the team is divided into specialized roles covering frontend development, middleware support, backend systems, and cross-platform engineering. Each team member plays a defined role to ensure efficient collaboration, iterative delivery, and technical reliability across both iOS and Android platforms.

This approach allows for agile development, clear task ownership, and strong communication throughout each iteration of the project lifecycle.



4. Document Control

This section outlines the tools and practices used by the Omnigym development team to manage project documentation, track version controls, and maintain collaboration across different platforms. It includes two key components:

1. **Document Control Tools** – A list of the main platforms and systems used to create, manage, share, and maintain project materials.

2. **Document Control Files** – A list of key documents and artifacts across project iterations, including their storage locations and associated versioning.

Document control provides transparency, accessibility, and consistency throughout the development lifecycle.

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

5. Management Representation

The Management Representative plays a key role in maintaining the quality, security, and user engagement of the Omnigym application. This role involves overseeing core quality processes, implementing improvements based on user and stakeholder feedback, evaluating technology choices, and ensuring that the app aligns with current industry and user-centered standards. Below is a breakdown of how the Management Representative contributes to sustaining and enhancing the app's overall quality framework.

5.1 Feedback Communication and Management

User Feedback Form

A user feedback option is integrated into the Omnigym app via the support page. Users are encouraged to submit suggestions to improve the app, such as feature requests, usability improvements, or bug reports. This open communication channel promotes community-driven development and engagement.

Feedback Review and Responses

The Management Representative regularly reviews user-submitted feedback. Key insights are analyzed to identify usability issues, feature requests, and enhancement opportunities — especially those related to community engagement, event participation, workout tracking, and leaderboard functionality.

Communicating Feedback to the Team

Collected feedback is organized and shared with the development team based on priority. This communication takes place during sprint meetings and via GitHub Discussions. The MR facilitates collaborative decisions regarding UI updates, performance improvements, and new feature implementations.

5.2 User-Focused Quality and Standards

Quality Policy Oversight

The Management Representative will ensure that the Omnigym team develops and implements updates with the quality policies in mind. The main focuses being on a secure, updated gym member verification system, responsive design, and easy and engaging user experience.

Updated Security Standards

The MR verifies that privacy and security best practices are being followed. This includes secure authentication, data compliance via Supabase, and responsible social platform features (e.g., public/private profiles, content moderation, admin approval workflows). All standards are assessed against current mobile and fitness-tech industry expectations.

Agile Updates and Responses

As the application evolves, the MR is responsible for maintaining quality across development cycles. This includes tracking iteration feedback and supporting decisions related to UI redesigns, new feature rollouts, and ongoing security enhancements.

6. 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 clean documentation at every step. This not only creates effective communication among end-users, stakeholders, and developers but also establishes trust and accountability. In the future, Omnigym 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, having a dynamic, interactive, and safe community platform.

7. Disaster Recovery/Backup Plan

7.1 Objectives

- **Minimize Downtime:** Quickly restore development or production environments in the event of system crashes or data corruption.
- **Protect Data Integrity:** Maintain safe and reliable backups of source code, documentation, and databases.
- **Improve Resilience:** Use retrospective insights to adapt tools and workflows for better fault tolerance.

7.2 Disaster Recovery Procedures

Code Recovery

- **GitHub Version Control:** All code is stored in a GitHub repository with multiple branches for pull request and merge control.
- **Revert Functionality:** GitHub's commit history allows recovery from unintended changes using revert and branch restoration options. It also allows for setting restrictions so commands such as force merge are prohibited to prevent damaging the main branch.
- **Backup Branches:** the team is encouraged to create pre-merge backup branches when pushing major updates.

Database Recovery

- **Supabase Snapshots:** Automated daily snapshots of the PostgreSQL database allow reverting back in case of scheme or data misconfigurations.

- **Change Tracking:** Any significant structural updates to the database are recorded and discussed in advance. There are also policy restrictions to prohibit certain changes depending on the assigned role of the developer.

Documentation and Assets

- **Microsoft SharePoint:** Project documentation is backed up in SharePoint with version history enabled. Comments can also be made and changes tracked in MS settings.
- **Figma Auto-Save:** Design assets are stored in Figma with full version control and change tracking. Hard copy backups are also saved by the designers to their computer storage.

Communication and Incident Response

- **Incident Documentation:** Any recovery event is logged in Microsoft Teams, communicated to the leaders, or GitHub Discussions/Issues for transparency.
- **Team Retrospectives:** Issues worth addressing are reviewed during scrum meetings and logged in the Meeting Minutes to discuss process breakdowns and improvement actions.

7.3 Backup Strategy

Asset Type	**Backup Location**	**Frequency**	**Managed by:**
Source Code	GitHub	Per Commit	All Devs and Reviewed by Violet
Database	Supabase Snapshots	Daily	Jawad and Violet
Documentation	SharePoint/MS Teams	Weekly	Jawad and Violet
UI/UX Designs	Figma (Cloud)	Continuous	Violet

7.4 Lessons Learned from Previous Iterations

During Iteration 1 and 2, the team encountered issues with GitHub merge conflicts when merging backend changes into the main branch. Notable problems included:

- **Branch Conflicts:** File renaming and structural changes caused merge instability and broken links in the code.
- **Dependency Conflicts:** Inconsistent versions across machines led to environment errors.
- **Environment Setup:** Local dependency mismatches caused breakdowns in testing.

Remedial Actions and Improvements:

- **Training:** Merge conflict resolution training and Git best practices are being introduced. Video shorts created by Violet are being shared through Teams to demonstrate common GitHub actions.
- **Backup Branches:** Team members now create backup branches before major merges.
- **Naming Conventions:** Standardized branch naming (e.g., /index.tsx, /config.py) improves traceability.

By combining structured disaster recovery processes with continuous learning from past challenges, the Omnigym team is building a more resilient and quality-driven development lifecycle. The Disaster Recovery and Backup Plan will be reviewed periodically and updated as needed to ensure project continuity and minimize disruption.

7.5 User Related Disaster Recovery Procedures

User Account Recovery

Role-based access control and daily backups ensure that all user account data is safely kept in Supabase. If anything is inadvertently removed or altered without authorization:

- **Email Verification and Reset**: Automated Supabase Auth procedures allow users to reset passwords or validate accounts. To guarantee dependability, these are checked on a regular basis.
- **Activity Logs**: To track and audit user-related modifications for questionable or unintentional action, Supabase logs are utilized.
- **Manual Restore**: In critical cases, project leads can manually restore user records from the most recent backup snapshot.

Session and Authentication Failures

To reduce user disruption from token errors:

- Real-time validation: Incorporated into the login form to guard against common user input problems including missing fields or invalid email types. Errors are prominently presented along with correction suggestions.
- Forgot Password Flow: The "Forgot Password" option is available to users who are unable to log in because they have forgotten their password. This enables them to swiftly restore access by sending a secure password reset email generated by Supabase.
- User Feedback: Rather to displaying generic failure messages, each authentication issue prompts the user with helpful feedback, such as "Email not found," "Incorrect password," or "Account not verified yet," which helps them find a solution.
- Auto-Logout Handling: When a token expires, users are asked to reauthenticate as soon as feasible to avoid losing their data.

User Communication During Disruptions

In the event of a user-facing incident:

- Banner Alerts: Using feature flag toggles, time-sensitive problems (such as outages or degraded features) cause an in-app alert banner to appear.
- Direct Email Notifications: In the event of a major incident, a system-generated email detailing the problem and the estimated time of resolution is sent to the impacted users.

Team Response The development team assigns and tracks recovery activities in GitHub Issues and Microsoft Teams after triaging incidents. During retrospectives, critical events are examined, and processes are modified as necessary.

The team makes sure that disruptions are promptly fixed and user trust is preserved by anticipating user-side problems in advance and incorporating clear recovery pathways and communication techniques.

8. Requirements Gathering

Before we began designing the Omnigym Social App, we took time as a team to figure out exactly what this app needed to do and who we were building it for. In addition to drawing from our own gym experiences and common frustrations with fitness apps, we also conducted a few informal surveys by speaking directly with gym members in our communities. Through these conversations and team brainstorming sessions, we were able to identify real needs, preferences, and missing features in the current gym app landscape. **Our goal was to design an app that brings people together through friendly competition, personal progress tracking, and social engagement—making the gym feel more motivating and community-driven.** With that goal in mind, we shaped our apps requirements around the feedback we gathered and the real habits we have observed at the gym.

8.1 Target Audience

We started by identifying the main people and stakeholders who would use or benefit from our app. This helped us focus our ideas on what really matters for each group:

- **Gym Members:** People who want to track workouts, compete with friends, meet new people, and stay motivated and accountable.
- **Personal Trainers:** Trainers who need better ways to connect with clients and track progress.
- **Gym Staff:** Employees and managers who handle day-to-day operations, events, and member data.
- **Gym Owners:** People focused on increasing engagement, membership retention, and revenue.

- **Our Development/Support Team:** We also considered our own needs as developers—like keeping the app stable, secure, and easy to update.

8.2 Requirement Considerations

To gather the requirements for Omnigym, we used a combination of informal surveys and internal team discussions. A few of our team members spoke directly with gym-goers in their communities to ask what they would want in a fitness app, what motivates them, and what tends to discourage them from sticking with their routines. At the same time, we held open brainstorming sessions as a team, pulling from our own experiences and frustrations with workout tracking and social gym apps.

We focused our conversations around simple, practical questions like:

- "What would make you actually want to open a gym app every day?"
- "What keeps you motivated at the gym—and what causes you to fall off?"
- "How can we reduce the awkwardness people feel when they first start going to the gym?"
- "What tools would make life easier for trainers or gym staff?"
- "What features would make the gym feel more social and connected?"

From both the feedback we received and our own insights, we shaped a feature list focused on friendly competition, progress tracking, and community building. These became the foundation for our app's user stories, wireframes, and technical design.

8.3 Key Feature Decisions

After several rounds of discussions and utilizing different tools like Edotor and Figma for brainstorming and diagrams, we ended up with a set of key features that felt meaningful, realistic, and valuable to users. Some of the major *functional requirements* we focused on included:

- **Secure User Accounts**
    - Signup and login with gym membership verification
    - Customizable profile with privacy options
- **Workout Logging & Tracking**
    - Ability to log workouts, track progress, and record personal records
    - AI-based workout recommendations (paused development due to time constraints)
- **Social Features**
    - Messaging between members and trainers
    - Leaderboards to showcase PRs and encourage competition
    - Group and event pages to bring people together
- **Role-Based Access**
    - Admins can verify memberships and manage gyms
    - Trainers can view client data and communicate directly

We also set some key *non-functional requirements* to make sure the app performs well and is built to last:

- Smooth and responsive experience on both iOS and Android devices.
- Secure handling of user data, including encrypted authentication.
- Scalable backend that can grow as we add more gyms and databases.
- Simple, easy-to-navigate UI for all types of users.

8.4 Feature Prioritization

Since we couldn’t build everything all at once, we prioritized features based on:

- What would have the biggest impact for users early on?
- What was realistic to complete within our Iteration 1 and 2 timelines?
- Which features depended on others working first?
- What our stakeholders (i.e., staff and gym owners) would find most valuable?

We made sure to start with things like login, basic navigation, and profile pages before layering in social features, messaging, or AI. As we continue into Iteration 3, we’ll finish building out the backend and move forward with the more advanced features that rely on this core foundation.

8.5 Requirement Gathering Log

Log Type	Tool/Location	Details
Brainstorming Sessions	MS Teams, Edotor	Notes taken during team meetings to capture early feature ideas.
Survey Notes	MS Teams	Informal interviews conducted with gym-goers across multiple communities.
User Story Board	MS Sharepoint, Presentation PowerPoint notes	User stories and tasks categorized and assigned during early planning.
Prototype Designs	Figma	Linked Frames and UX notes for key pages and features. Utilized color theory concepts into app designs and logo creations.
Technical Requirements	GitHub Discussions	Feature breakdowns and backend requirement discussions.

9. Code Review

9.1 Procedure

The purpose of code reviews in the Omnigym project is to maintain code quality, consistency, catch bugs early, and promote team collaboration. Regular peer reviews help enforce best practices, share knowledge and experience that may help others in their development, and reduce the risk of introducing technical delays.

Review Process

Step	Description
1. Create a Pull Request (PR)	Developers push changes to the main branch and open a pull request to main.
2. Assign Reviewers	If leader is unavailable, at least one other team member is to be contacted to review the code either through GitHub or sharing screens on MS Teams.
3. Reviewer Responsibilities	Check for code clarity with comments and a walk-through of functionality with developer. Inspect merge conflicts and potential code misconfigurations that may have been missed.
4. Request Changes/Approve	Reviewer can approve the pull request or request that changes be made. Feedback is given via GitHub comments or Teams messaging.
5. Final Merge	Once approved, the code can be merged into the main branch by the developer or lead.

Review Standards

Reviewers are encouraged to evaluate code based on the following:

- **Functionality:** Does the code do what it's supposed to?
- **Readability:** Is the code clean, well-commented, and easy to understand?

- **Efficiency:** Is the logic optimized with concepts like encapsulation and scalable?
- **Security:** Are there any vulnerabilities or exposed endpoints that need to be removed like security keys? Are *.gitignore* files properly working to prevent unwanted files being pushed (i.e., .env file).
- **Consistency:** Does it follow Omnigym's brand identity, code style, and folder structure?
- **Testing:** Are edge cases handled during error handling? Is test coverage appropriate?

Tools Used

- **GitHub Pull Requests** – For formal review, Git Guardian checker, inline commenting, and version tracking
- **VS Code Extensions** – For formatting, linting, and debugging
- **GitHub Projects/Issues** – For linking reviews to specific tasks or features

Roles and Responsibilities

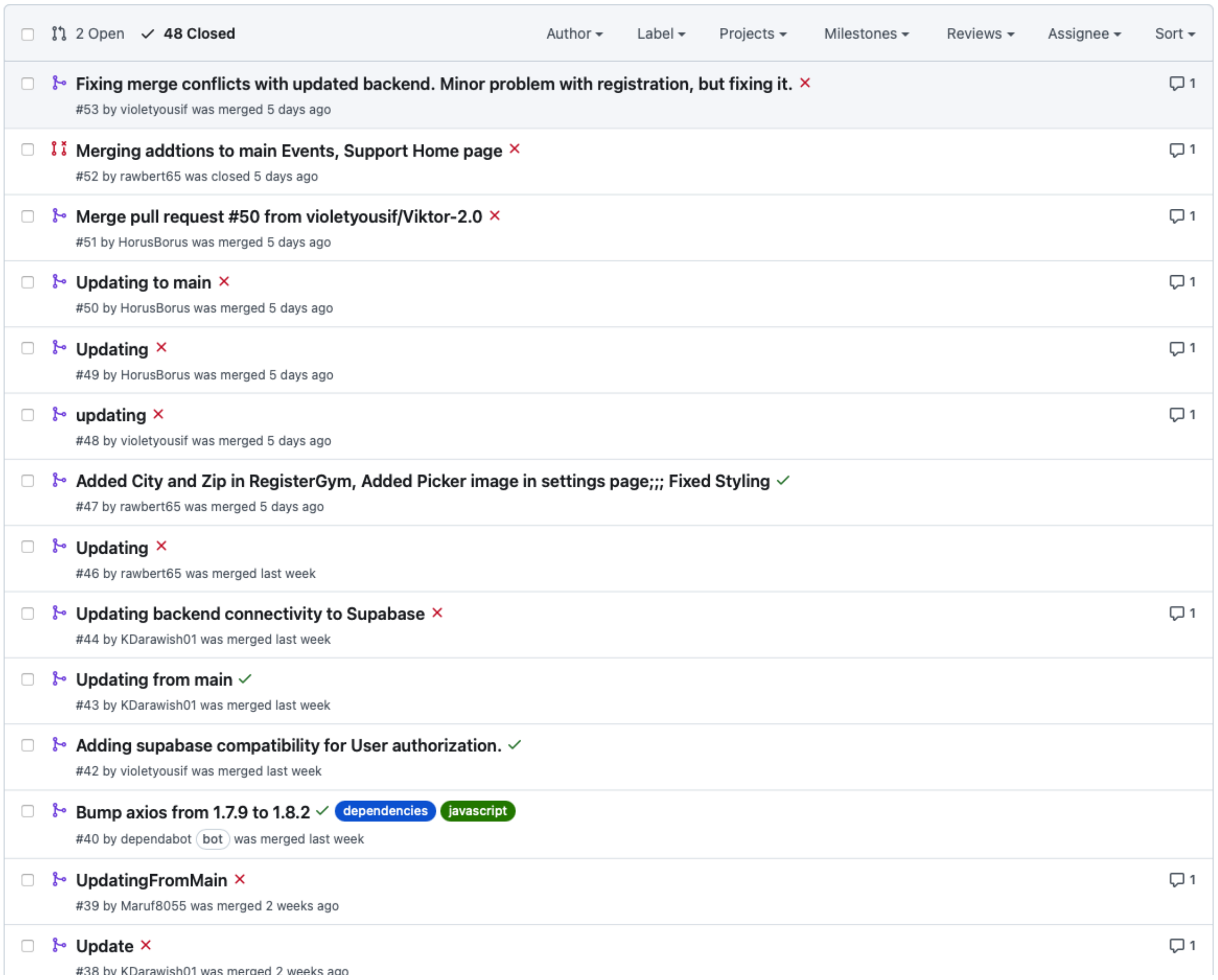
- **Reviewer(s):** Leaders or team members assigned to review code based on feature ownership or area of expertise.
- **Developer (Pull Request Author):** Responsible for submitting clean, tested code with appropriate documentation.
- **Project Lead:** Approve/override pull request decisions for critical path features.

Review Frequency

- Code is reviewed as part of each pull request before merging into main branch.
- All major features and backend logic require at least one approval before integration.

9.2 Code Review Log

Below is a snapshot preview of our code review log. Most codes were reviewed using Teams with other members and/or Leaders. Unfortunately, many commits appear to have conflicts due to a dependency issue in CodeQL that is being addressed by their development team.



2 Open ✓ 48 Closed

Fixing merge conflicts with updated backend. Minor problem with registration, but fixing it. ✗ 1
#53 by violetyousif was merged 5 days ago

Merging addtions to main Events, Support Home page ✗ 1
#52 by rawbert65 was closed 5 days ago

Merge pull request #50 from violetyousif/Viktor-2.0 ✗ 1
#51 by HorusBorus was merged 5 days ago

Updating to main ✗ 1
#50 by HorusBorus was merged 5 days ago

Updating ✗ 1
#49 by HorusBorus was merged 5 days ago

updating ✗ 1
#48 by violetyousif was merged 5 days ago

Added City and Zip in RegisterGym, Added Picker image in settings page;;; Fixed Styling ✓
#47 by rawbert65 was merged 5 days ago

Updating ✗
#46 by rawbert65 was merged last week

Updating backend connectivity to Supabase ✗ 1
#44 by KDarawish01 was merged last week

Updating from main ✓
#43 by KDarawish01 was merged last week

Adding supabase compatibility for User authorization. ✓
#42 by violetyousif was merged last week

Bump axios from 1.7.9 to 1.8.2 ✓ dependencies javascript
#40 by dependabot bot was merged last week

UpdatingFromMain ✗ 1
#39 by Maruf8055 was merged 2 weeks ago

Update ✗ 1
#38 by KDarawish01 was merged 2 weeks ago

For the full closed pull request history, please navigate to the GitHub Repository Pull Requests page or *https://github.com/violetyousif/OmniGymSocialApp/pulls?q=is%3Apr+state%3Aclosed*.

10. Feasibility

The following tables outline the feasibility of key features, components, and technologies used in the Omnigym Social App. Each item has been assessed based on current implementation status, technical constraints, and future development plans.

10.1 Feature Feasibility Log

Feature	Description	Feasible? (Y/N)	Rationale/Notes	Action Required
Gym Membership Verification	Verify users with gym databases (Planet Fitness, Lifetime Fitness) using AffilGyms + memberID	Yes	Custom models + Django ORM working; Supabase setup complete	Ongoing testing with edge cases
Two-Step Registration Flow	Step 1: Verify gym info → Step 2: Create user account	Yes	Successfully implemented in React Native with validation	Improve user feedback between steps
Gender Dropdown & DOB Picker	Form validation and styled picker inputs for iOS and Android compatibility	Yes	Works in Expo/React Native; UX refined	Monitor for platform and dependency inconsistencies
Terms & Privacy Modal	Modal popup before account creation	Yes	Modal is implemented and functioning with Date updates	Link to live policy docs
React Native Frontend with Expo	Mobile app frontend	Yes	Stable setup and development in progress	Monitor Expo updates
Django Backend (REST API)	Handles input data, DB queries, validations, error handling	Yes	Refactored to support unified schema and	Test for speed and scalability

			gym-specific logic	
User Login / Auth Token System	Secure login with JWT/session tokens	In Progress	Design planned, not yet fully implemented	Choose between JWT or Django session auth
User Dashboard	View workout logs, gym stats, social posts	Not yet	Design planned, not yet fully implemented	Choose between JWT or Django session auth
User Dashboard	View workout logs, gym stats, social posts	Not yet	Planned for future phase	Define scope and wireframes
Friend / Social Connection System	Users can follow, message, and react	Not yet	Core idea mapped, no backend or UI yet	Define schema and interaction logic
Notifications / Reminders	Push notifications for gym goals or friends	Not yet	Not researched yet	Explore Expo push notifications or Firebase
Cloud Database Sync (Supabase)	Sync app data in real-time	Yes	Supabase integration active and reliable	Continue optimizing queries; merge error with main needs to be fixed though
Admin Tools for Gym Managers	Gym-side tools for user management	Not yet	Could be added as a separate admin app	Research admin scope and permissions

10.2 Technical Stack Feasibility Log

Component	Status	Notes
React Native + Expo	Integrated	Dev environment works well. Restructured project to fit conventional directory and file naming
TypeScript	Integrated	Enforces type safety in frontend
Django + Django REST Framework	Integrated	Backend refactored to support gym logic
PostgreSQL (via Supabase)	Integrated; In Progress	User tables functional
Supabase Authentication	Integrated	Using custom auth flow instead
SQLite3/Flask	Deprecated	Switched to Supabase/PostgreSQL
Ionic + React	Deprecated	Too many limitations and learning curves with mobile development. Switched to React Native + Expo
Firebase	Deprecated	Too many limitations with relational data tables. Switched to Supabase.

Pending Research Items

- How to implement inbox communication
- Pros/cons of using Django's built-in auth vs JWT
- Offline-first strategies for mobile
- Secure storage of tokens or session data in Expo
- AI implementation for routine creation

11. Scrum/Sprint Meetings

The purpose of sprint meetings in the Omnigym project is to organize development efforts into manageable iterations, maintain team alignment, and ensure continuous improvement. These meetings are used to plan for upcoming tasks and deadlines, track progress, address issues, and reflect on outcomes. Our typical structure involves 1-2 meetings a week led by Violet and co-led by Jawad. The Sprint Meeting minutes template can be found in Appendix A of this document, while the records can be found in the most updated Iteration documentation.

12. Ensuring Collaboration

Efficient and seamless collaboration was an underlying facet of the Omnigym Social App project, and the team employed various mechanisms and tools to ensure collaborative and productive collaboration. Below is how collaboration was guaranteed, with some specific input from team members like Jawad and Violet.

12.1 Agile-Scrum Methodology

The team employed an Agile-Scrum framework to facilitate iterative development and continuous improvement. This ensured that activities were broken down into manageable sprints, with regular reviews and retrospectives to keep up with stakeholder feedback and adapt to changing requirements.

12.2 Task Management Using MS Planner

Jawad played a key role in introducing Microsoft Planner for the Team. This tool was used to:

- **Delegate Tasks:** Identify and assign tasks to specific members based on their capacity and line of duty.
- **Monitor Progress:** Monitor the progress of tasks and determine if deadlines were met.
- **Coordinate Actions:** Coordinate and facilitate communication between the members to avoid overlaps and effort gaps.

12.3 Role Clarity and Delegation

Violet, who was the UI Designer, Full Stack and Executive Lead, ensured roles and responsibilities were assigned and established as follows:

- **UI/UX Design:** Representative of design work and ensuring the user interface was intuitive and in line with user specifications.
- **Task Oversight:** Ensuring task distribution and ensuring that each team member understood what they were assigned to do.
- **Progress Monitoring:** Regular check-ins with teammates to confirm that work was on track and working through blockers.

12.4 Communication Tools

The primary communication tool used by the team was Microsoft Teams to establish:

- **Real-Time Communication:** Stand-up meetings and impromptu conversations to address urgent matters.
- **Document Sharing:** Shared access to project documents, meeting minutes, and design documentation.
- **Transparency:** Open channels of feedback and collaboration between all members.

12.5 Team Meetings and Collaboration Spaces

- **Weekly MS Team Calls:** Weekly team calls were conducted to discuss progress, challenges, and project goals. Individual calls were also made throughout the week for updates and one-on-one assistance.
- **Conference Rooms:** In-person meetings took place in the ATEC Conference rooms so that discussions and closer collaboration could be facilitated. These proved to be most useful for solving complex issues and for fostering team spirit.

12.6 Version Control with GitHub

GitHub was used for version control and log records, allowing the team to:

- **Track Changes:** Maintain an audit trail for changes to code and contributions.
- **Collaborate on Code:** Use pull requests (PRs) for reviewing and merging code so that all changes were reviewed prior to integration.
- **Resolve Conflicts:** Manage merge conflicts in an ordered way, as documented in the lessons for Iteration 1.

12.7 Design Collaboration with Figma

Figma was used for UI/UX design and prototyping so that the team could:

- **Design Together:** Real-time collaboration on design elements and user flows.
- **Get Feedback:** Share prototypes with stakeholders and incorporate feedback incrementally.
- **Ensure Consistency:** Check design elements were consistent across the application.

12.8 Knowledge Sharing and Documentation

- **How-To Guides:** Violet also created detailed how-to guides for various sections of the project, such as setup guides, design documents, and troubleshooting tips. The guides were uploaded to Microsoft Teams channels so that everyone on the team could access the information they needed.
- **Centralized Documentation:** Have an entire project wiki and README files so that everybody on the team had access to project information.
- **Knowledge Sessions:** Hold tutorials and knowledge sessions to keep everyone aligned on project goals and technical details.

12.9 Regular Retrospectives

The team held regular sprint retrospectives to:

- **Reflect on Progress:** Discuss what worked and what didn't, and how to do it differently.
- **Identify Bottlenecks:** Fix issues and streamline processes for future sprints.
- **Celebrate Wins:** Celebrate team achievement and maintain morale.

With the deployment and implementation of these strategies and also using tools like Microsoft Planner, GitHub, Figma, and Microsoft Teams, the team was able to enable productive collaboration, open task allocation, and an agreed workflow. Through this structured yet flexible approach, the team could provide quality features and uphold transparency and accountability during the project life cycle.

13. Elicitation of Feedback

Throughout the process of developing the Omnigym Social App, the team gathered feedback from stakeholders such that the final product would satisfy the user needs and expectations. The feedback was gathered through a variety of methods, including:

- **User Surveys:** Undertaken among gym members to validate the idea of the app and receive preliminary feedback.
- **Interviews:** With gym owners, trainers, and staff members to understand their unique needs and challenges.
- **Prototype Testing:** Elected to share initial designs with stakeholders to solicit iterative feedback and iterate on features.
- **Sprint Reviews:** Regular show-and-tell of progress to stakeholders for continuous alignment and realignment.

13.1 Feedback Sources

Source	Type of Feedback	Tool Used
Gym Members/App Users	Usability, feature suggestions, bugs	In-app feedback and support form, verbal interviews
Personal Trainers and Gym Staff	Functionality, accessibility, usefulness	In-app feedback form, informal surveys
Internal Team Members	Technical issues, process improvements	Scrum meetings, GitHub Discussions, Teams messaging
Mentors/Peers	Quality feedback, milestone iteration review input	Class presentations, emailing

13.2 Feedback Methods

- **Informal User Interviews:** Conducted by team members with friends or gym-goers in their local communities.
- **In-App Feedback Form:** A planned feature to allow real-time user submissions (not ready for release yet).
- **Sprint Retrospectives:** At the end of each sprint, the team holds a discussion to reflect on what worked and what could be improved.
- **GitHub Discussions and Issues and MS Teams:** Used for posting enhancement suggestions or bug reports tied to specific tasks.

13.3 Feedback Review Process

- Feedback is collected throughout development and reviewed during sprint planning and team meetings.
- Suggestions are evaluated for feasibility, relevance, and alignment with current development priorities and time restraints.
- Actionable items are turned into Planner tasks, GitHub issues, or added to the feature backlog.
- A developer or lead is assigned to follow up on each item through MS Planner.

13.4 Frequency

- Feedback reviews happen during sprint meetings**,** with urgent issues prioritized.
- Retrospectives are held at the end of every iteration to evaluate what went well and what needs to change.

13.5 Outcome

The app was developed to meet the needs of a wide range of stakeholders thanks to this proactive approach to feedback. The user-centric design that resulted enhanced operating efficiency and raised community participation in the gym.

13.6 Feedback Elicitation Log

Feedback Source	Feedback Type	Summary	Action Taken	Status
Gym Member (informal)	Usability	Suggested using the scoring system from professional weightlifting competitions	Researched different algorithms. Decided on the Wilks 2 score.	In Progress
Sprint Retrospective	Internal Process	Merge conflicts are slowing down progress.	Added Git conflict training and Violet is continuously working with the team to make them more fluent.	In Progress
Trainer (informal via instagram)	Feature Request	Requested calendar view for events	Discussed with team.	Logged
Design Lead	Visual Designs	Suggested brand color and logo implementation edits.	Updated theme in Frontend folder to reflect brand identity.	Completed
Tech Leader and Peers	Framework	Limitations with React and Ionic caused issues during first iteration.	Jawad tested React Native instead and a peer suggested using Expo with it for smoother mobile implementation.	Completed
Gym Member (informal)	Usability	Suggested a location on the profile to list more details about the member like relationship status.	Discussed with team. Still in consideration.	Logged

Team Lead	Internal Process	Integrating Firebase continued to be an issue due to its relationship database limitations and implementation restrictions.	Switched to Supabase to host app database.	Completed

13.6 Feedback Loop / User Acknowledgment

Demonstrating to users that their input results in action is crucial to preserving their trust. Because of this, the group put in place a simple feedback loop:

- Stakeholder-recommended changes are shared in team updates on MS Teams and during sprint reviews.
- Internally, labels and comments mentioning the feedback's source are included in GitHub Issues and Planner tasks.
- Future modifications will be shown in the app's changelog section, with tags like "User Suggested" to clearly acknowledge user efforts.

13.7 Future Feedback Plans / Scalability

The feedback procedure will change as the app expands to accommodate more user input and rising demand. Planned enhancements include:

- Automated Feedback Tagging: Using tags like "bug," "UX," or "feature," submissions made via the in-app form will be sent to GitHub Issues through a Zapier integration.
- Admin Dashboard Filters: Team leads will be able to arrange feedback according to frequency or criticality in a future admin view.
- Community Voting: To help prioritize community-desired changes, a voting method may be used to surface high-volume feature requests.

13.8 Feedback Prioritization Criteria

A straightforward prioritization system was employed to make sure the team concentrated on the comments that were most important. Every item was evaluated according to:

- Urgency: Is it preventing important features or creating serious problems?
- Impact: Will the modification significantly enhance system performance or user experience?
- Effort: Can the feedback be resolved quickly, or does it call for a large investment of resources?
- Stakeholder Value: Did a core user group—such as regular users or gym employees—provide the feedback?

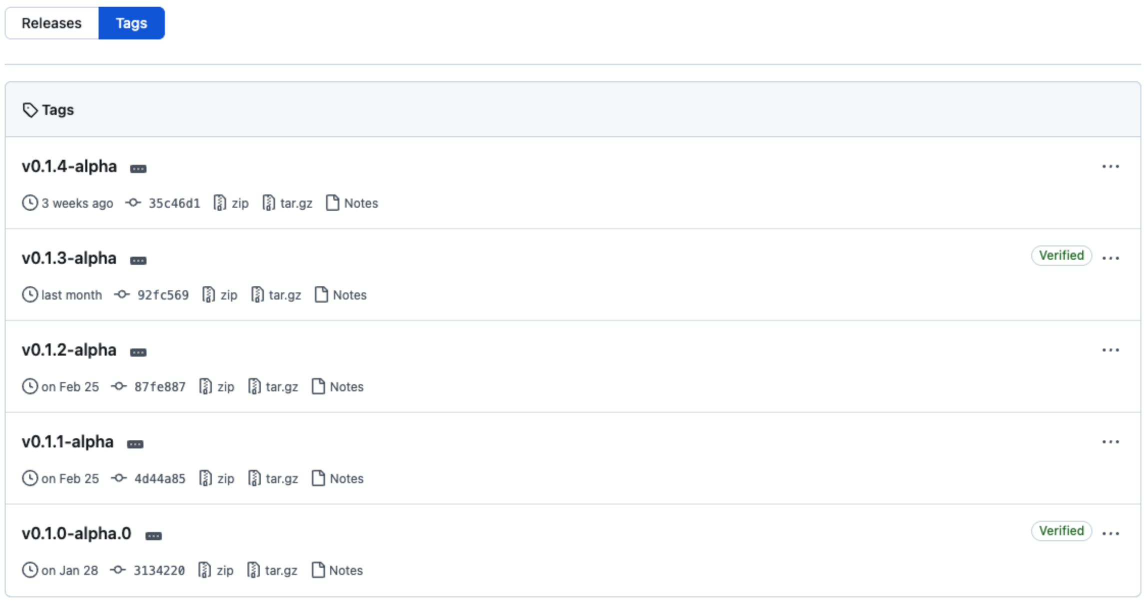
This made it easier to respect stakeholder feedback while balancing what could really be completed in each sprint.

14. Record-Keeping

Throughout the development of the Omnigym Social App, we kept detailed records to stay organized, accountable, and on track with our goals. With a fast-moving, collaborative project like this, it was important that every team member could look back at previous meetings, understand what decisions were made, and clearly see what tasks were assigned each week. Record keeping played a key role in helping us communicate effectively, adjust plans when needed, and reflect on our progress during sprint retrospectives.

To manage our documentation, we used a few different tools:

- **Microsoft Teams** for weekly meetings and coordination
- **MS Planner** for assigning tasks and tracking progress
- **GitHub** for version control, releases/tags, and commit logs



Releases | Tags

Tags

v0.1.4-alpha
3 weeks ago · 35c46d1 · zip · tar.gz · Notes

v0.1.3-alpha · Verified
last month · 92fc569 · zip · tar.gz · Notes

v0.1.2-alpha
on Feb 25 · 87fe887 · zip · tar.gz · Notes

v0.1.1-alpha
on Feb 25 · 4d44a85 · zip · tar.gz · Notes

v0.1.0-alpha.0 · Verified
on Jan 28 · 3134220 · zip · tar.gz · Notes

- **Figma and Edotor** for design records and feature planning

- **Scrum Meeting Forms** (in Word format) to document each sprint

Each week, the Scrum Master filled out a standardized meeting form that included the sprint date, time, attendees, achievements, backlog items, sprint goals, and a retrospective. These forms allowed us to look back and quickly understand what each sprint accomplished and where we could improve. The retrospectives were especially useful for identifying recurring challenges, like uneven task distribution or scheduling conflicts, and figuring out how to address them as a team.

Some key records from our meetings include:

- **Week 1:** Finalized the Omnigym app concept and created the Edotor feature graph
- **Week 2:** Completed rating system and decided on app name and logo. Prepared presentation and notes
- **Week 3:** Created GitHub repository and reassigned scrum master
- **Week 4:** Created Starburst points mockup
- **Week 5:** Decided on team roles and began programming frontend and backend
- **Week 6:** Completed Iteration 1 and presented to Forestview
- **Week 7:** Switched frontend to React Native + Expo and Backend to Django
- **Week 8:** Frontend was able to run successfully
- **Week 9:** Completed frontend registration
- **Week 10:** Switched databases from Firebase to Supabase
- **Week 11:** Successfully connected the Planet Fitness database to the backend
- **Week 12:** Focused on completing Full Stack integration and prepping for Iteration 3
- **Week 13:** Updated and completed Iteration 2 Documentation. Started Iteration 3
- **Week 14:** Completed setup of Supabase RLS policies and fixed tables
- **Week 15:** Completed Iteration 3 Documentation

All of our sprint records can be found in **Appendix B** of Iteration 3's documentation, which includes full meeting minutes for each week. These records helped us reflect on our progress, hold each other accountable, and continuously improve how we worked together as a team.

15. Vendor Compliance

Vendor compliance for the Omnigym Social App project includes the procedures and practices used by third-party vendors, services, and tools used during the application development and deployment to address the security, quality, legal, and operational requirements of the project. This includes development frameworks, design tools, GitHub repositories, database hosts, and communication platforms. The same should be accomplished to guarantee the integrity, reliability, and scalability of the application and eliminate external dependency-based risks.

15.1 Key Components of Vendor Compliance

Vendor Selection and Grading Criteria:

Vendors graded by reputation, technical competence, following the best practices relevant to their business line and ability to fulfill a project's needs.

VENDOR USE CASES:

- **Supabase:** Chosen due to having open-source (MIT), PostgreSQL-database-based database management, real-time processing, and tightly secure authentication capabilities.
- **Django:** Chosen due to security features, scalability, and backend app compatibility.
- **React Native + Expo:** Under the MIT license; same codebase for cross-platform app development as well as fast deployment.
- **GitHub:** Carries a free-tier license type and allows for source control and pull requests.
- **Microsoft Planner:** Enterprise access through school that allows for task tracking and delegating tasks and collaboration.

2. Contractual Requirements

- **SLAs (Service Level Agreements):** Vendors will be held to performance levels as stated (e.g., response time, uptime).
- **Data Protection Clauses:** An attempt to ensure that vendors use encryption standards and data privacy acts. For example, services handling user data (like Supabase) must align with data protection regulations such as GDPR.
- **Audit Rights:** Facilitates regular auditing in verifying terms of agreement compliance.

3. Compliance with Regulations

- **Data Privacy:** The vendors must comply with the applicable laws, most specifically user-related information.
- **Security Standards:** Cloud vendors (e.g., Supabase) must comply with industry-standard security standards (e.g., OAuth2, JWT).

4. Regular Monitoring and Audits

- Performance Metrics: Monitor the vendor's performance on a regular basis in terms of SLAs.
- Security Audits: Regularly securely audit the vendor system to detect and eliminate vulnerabilities.
- Update Policies: Have vendors offer regular updates and patches for services.

5. Risk Mitigation

- **Fallback Plans:** Develop fallbacks in the event a vendor falls out of compliance with compliance measures.
- **Multi-Vendor Strategy:** Avoid keeping all eggs in one basket and develop alternatives (e.g., Firebase as alternative to Supabase).

6. Documentation and Transparency

- Vendor Contracts: Keep physical copies of contracts, e.g., compliance terms.
- Compliance Reports: Monitor vendor compliance status and report to stakeholders for transparency.

7. Implementation in the Omnigym Project

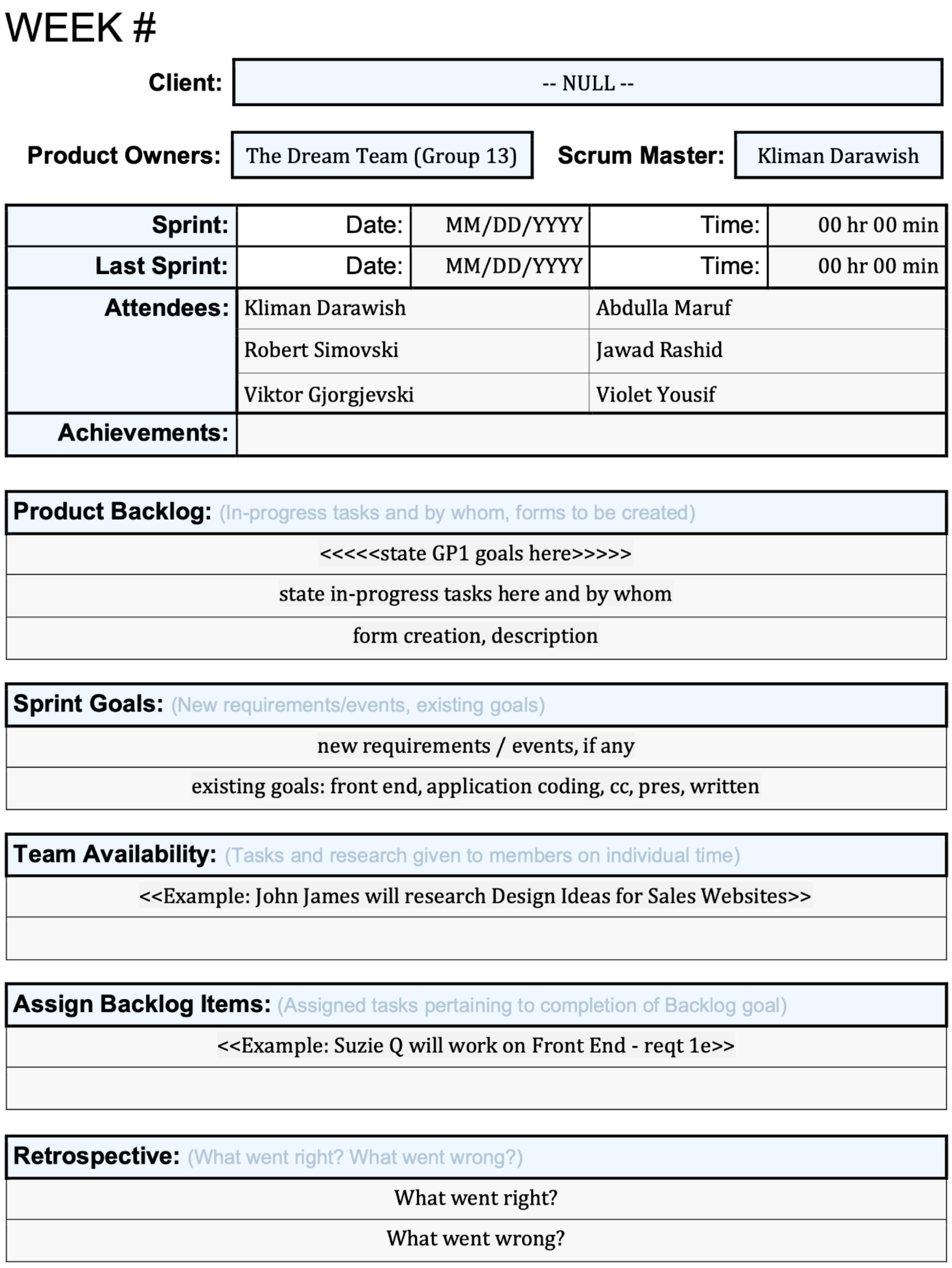
Vendor compliance in Omnigym Social App is regulated by:

- **Supabase:** PostgreSQL database best practice compliance, encryption policy compliance, and real-time data synchronization compliance standards.
- **Django:** Python application performance standards and PEP8 compliance.
- **React Native + Expo:** Framework compliance with cross-platform development and performance standards.
- **Third-Party Libraries:** Open-source library security vulnerability and compliance testing.

Omnigym's team assists with vigilant vendor compliance management that every third-party dependency fulfills project goals, safeguards user information, and assists in maintaining the app stable and scalable. It reduces risk and costs nothing to long-term viability.

Appendix A

Scrum Meeting Minutes Template:



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?