3 Project Plan

3.1 Project Management/Tracking Procedures

The project management style that is best suited for this project is agile. Bi-weekly meetings have been set up with our advisor and client. Agile works well for us to be able to run two-week sprints and then demo what we have completed in those two weeks the next time we meet with our advisor and client. Agile allows us to take an iterative approach to developing both iOS and Android applications. We don't need to be held up for another part of the project because it has been broken down into specific features that anyone can start working on without waiting for another part of the project first. Lastly, it allows us to measure our progress as we go, rather than waiting for a specific milestone to be hit.

In order to track our progress throughout the course of this and the next semester, our team is utilizing Git. A GitLab has been created for us to track the project. We have created issues for each component of the project so that when someone is ready, they can move the issue into the developing column of our issue board. Once completed, we assign another team member to code review the issue before merging to our main branch. Our advisor and client also have access to GitLab to view our design documentation and track our progress too.

3.2 Task Decomposition

The tasks for our project are broken down into 3 tasks: iOS development, Android development, and security solutions. The development sections include the same subtasks to create the application, develop the screens for the project, connect the screens together, connect the application to the existing backend, connect to the physical rack, and testing. The security section included the subtasks to develop cybersecurity solutions, research industry standards and requirements, integrate solutions into the application, integrate solutions into the database, and penetration testing (See figure 3.2.1).

Not all of these subtasks are dependent on the ones before it. For example, screens can be connected together before all the screens are created. This allows us to work in sprints to get working functionality for our client without being held back by small issues in specific subtasks or missing client information.

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

First semester milestones:

Screen designs are implemented following the user interface designs described by the client's documentation and demonstration; 100% of the screens will be implemented and viewable in both iOS and Android applications.

Navigation between screens is implemented aligning with the client's documentation; 100% of the screens will be accessible through the navigation flow of the iOS and Android applications.

Second semester milestones:

Cyber Security is successfully implemented to secure the backend database and remove any security vulnerabilities from the application; different users will only be able to access the data they are authorized to view.

The application can connect to the Force Rack over Bluetooth connection using different types of devices; the device will connect to the rack within 30 seconds, notifying the user if connection has failed.

3.4 Project Timeline/Schedule

Our Ghatt chart (See figure 3.4.1) shows the time in which each task is projected to be finished. However, the connection and implementation of the application may be extended into next semester due to bugs in the backend and database.

Below are the subtasks that are included in a Ghatt chart:

- **Task 1: Research**
  - Meet with professors and brainstorm - end of September
  - Research and familiarize with Android and IOS - mid-October
  - Pen test application and website to find connections in the product - beginning of October
  - Research applications to better secure database and two-step authentication - end of October
- **Task 2: Creating an Android application**
  - downloading and creating Android application - beginning of October
- **Task 3: Creating an IOS application**
  - downloading and creating IOS application - beginning of October
- **Task 4: Create Design screen layouts**
  - Create Designs in Figma and replicate the actions to connect the screens together - mid-September
- **Task 5: Set up the IOS device in the lab**
  - Log in to the computer device and set up ssh/establish connection to lab computer on personal computers - mid-September
- **Task 6: Create Screens**
  - Create IOS screens - mid-October
  - Create Android screens - mid-October
- **Task 7: Establish screen navigation**
  - Use Figma to create mock-up of screen navigation - End of September
  - Test navigation - beginning of October
- **Task 8: Connect screens with backend API hooks**
  - use API hooks give by client to connect to backend - End of March
  - Test Connection from frontend to backend - End of March
- **Task 9: Connect applications to workout rack**
  - Receive real time data from rack to application - Mid May
  - Display real time data - Mid May
  - Test different scenarios - Mid May
- **Task 10: Integrate security solutions into apps**
  - Create and deploy security solution into both iOS and android apps - End of May
- **Task 11 : Integrate security solution into database**
  - Create and deploy security solution into database- End of May

Due to our agile model, we also have the following sprint schedule

- **Sprint 1: Product Design (Start of September - Start of October)**
  - Meet with client and advisor to get stakeholder needs and required functionality
  - Decide how to approach project logistically
    - Team member assignment, concurrent/sequential app development, etc.
  - Create screen designs for system UI and navigation overview
  - Set up add development environment
- **Sprint 2: Screen Development and Security Features (Start of October - Mid December)**
  - Create dynamic screen in development environments according to UI designs
  - Establish navigation between all screens
  - Implement basic functionality of system
    - On-click events, editable text fields, etc.
  - Decide on security features to be implemented based on research and penetration testing
- **Sprint 3: Backend and Security Integration (Mid January - March)**
  - Create connection between frontend, backend, and database
  - Implement functionality based on backend data
    - Auto filling data fields, pushing data into the backend, etc.
  - Begin implementation of security features for database
- **Sprint 4: Hardware and Security Integration (Mid January - May)**
  - Create bluetooth connection between hardware rack and software
  - Implement live data updating between hardware and software
  - Begin implementation of security features for iOS and Android applications
  - Implement any missing app functionality
- **Sprint 5: Testing (February - May)**
  - Ensure correct app functionality
  - Test security gaps
  - Prepare for final product presentation

3.5 Risks And Risk Management/Mitigation

Lab Setup

- **Risk:** Unable to set up an iOS environment, board, and rack in the lab before testing
- **Likelihood:** Unlikely (0.2)
- **Consequences:** Major (Would not be able to test the app with live data)
- **Mitigation:** Ask the client for a rack and use existing or made up data.

Backend Connectivity

- **Risk:** Unable to get the API hooks from the current development team to connect our screen designs to the database and the hardware on the rack
- **Likelihood:** Unlikely (0.2)
- **Consequences:** Major (No data could be stored or retrieved making our project unusable)
- **Mitigation:** Alert the client of necessary database information needed so advisor and client can work towards delivering them to us.

Hardware Connectivity

- **Risk:** Unable to connect to the Force Rack through the Bluetooth stack, the team does not work on the hardware portion of the product
- **Likelihood:** Moderate (0.5)
- **Consequences:** Major (App would not receive any athlete data)
- **Mitigation:** Planned out minor testing in parallel with app feature development to frequently check the connection status as more functionality is added to the application, enabling the team to notify the advisor if there are issues.

Integrated Security Solutions

- **Risk:** Unable to identify and patch all security vulnerabilities
- **Likelihood:** Moderate (0.5)
- **Consequences:** Major (Leaves app and backend database vulnerable to attacks)
- **Mitigation:** Setup an intrusion detection system and two-factor authentication to mitigate as many security vulnerabilities as possible. Create an incident response plan for future security incidents.

3.6 Personnel Effort Requirements

Task	**Time (hrs)**	**Explanation**
Existing Application Research	1	This task involved the whole team working through the existing tablet application, noting improvements, studying the flow, and gaining an understanding of our application to be created.
Create Android Application	1	This task involves creating a new project on android studio and connecting the github to the project.
Create iOS Application	2	This task involves creating a new project on Xcode and connecting the github to the project. This task includes time to set up the emulator and learn how it worked. Our team has no experience with swift.
Design Screen Layouts	5	This task involves using figma to create a reference for the layout of every possible screen in both horizontal and vertical orientation based on the pre-set designs given by the client. This task also includes a flowchart of the navigation between screens.
Set up iOS development environment	2	Setup of project workstation in the senior design lab. This involves: MacOS setup, downloading Xcode and swiftUI playground, setting up remote access, receiving locker, tablet, and extra monitor.

Develop screens for Android	55	Creation of each screen as its own file with correct formatting for different orientations and sizing. Each screen development should be a separate git issue and branch. This includes merge conflict and other git issues.
Develop screens for iOS	60	Creation of each screen as its own file with correct formatting for different orientations and sizing. Each screen development should be a separate git issue and branch. This includes merge conflict and other git issues.
Connect Screens for Android	40	Addition of screen functionality into the project. This includes proper screen navigation based on user actions, simple features that don't require database information, and the skeleton code for backend implementation if possible. This includes merge conflict and other git issues.
Connect Screens for iOS	40	Addition of screen functionality into the project. This includes proper screen navigation based on user actions, simple features that don't require database information, and the skeleton code for backend implementation if possible. This includes merge conflict and other git issues.
Connect Android application to existing backend	20	Establishing the connection between the frontend and backend. Implementing the API hooks to call and push information to the database. Finishing code for functionality requiring backend. This will likely have a lot of problem solving as we can't see the backend code. This includes merge conflict and other git issues.
Connect iOS application to existing backend	20	Establishing the connection between the frontend and backend. Implementing the API hooks to call and push information to the database. Finishing code for functionality requiring backend. This will likely have a lot of problem solving as we can't see the backend code. This includes merge conflict and other git issues.
Connect Android application to hardware rack	20	Establishing the connection between the frontend, backend, and hardware so that data from the hardware rack is properly transmitted through frontend and backend. This will include research in regards to connection the rack with mobile devices as well as Bluetooth implementation research. This includes merge conflict and other git issues.

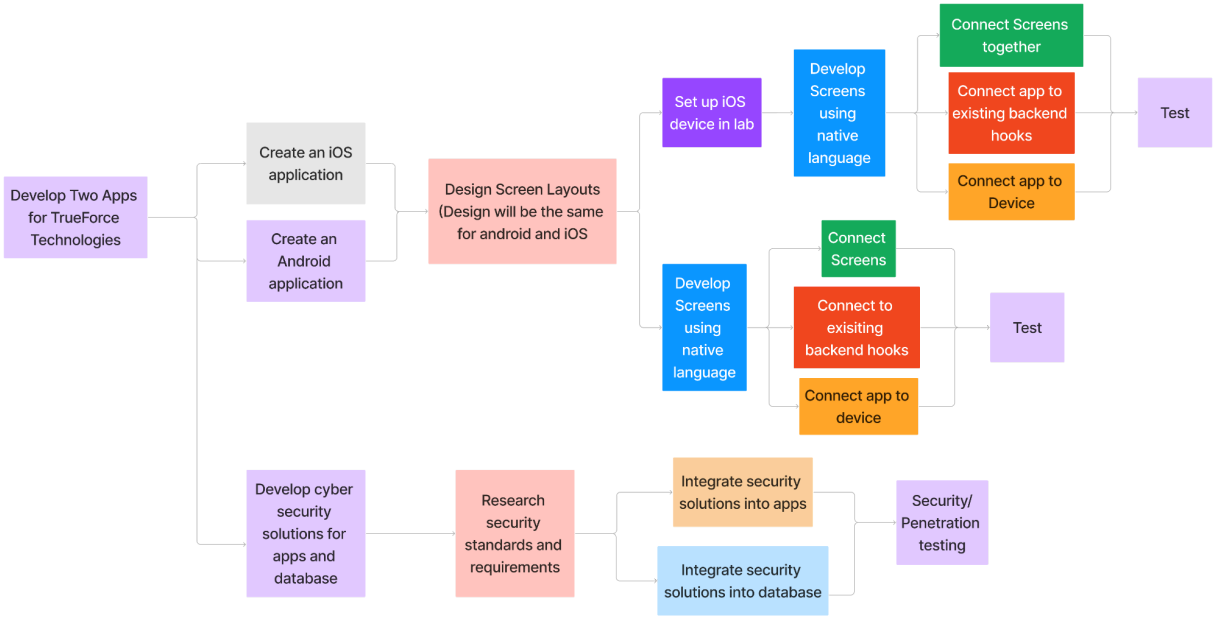
Connect iOS application to hardware rack	20	Establishing the connection between the frontend, backend, and hardware so that data from the hardware rack is properly transmitted through frontend and backend. This will include research in regards to connecting the rack with mobile devices as well as Bluetooth implementation research. This includes merge conflict and other git issues.
Application testing	80	Test both android and iOS apps to ensure correct functionality. Things to test include for example: rigorous navigation, improper data inputs, attempts to "break" the system, and graph scaling changes. This includes fixing code when problems are found and git issues.
Develop Security Solutions	40	Design possible security features for both Android and iOS applications and the database. Includes meeting with security professors to determine applicable solutions and feature research. Communicate with clients about the best course of solutions and find solutions within budget.
Research Security Standards & Requirements	50	Investigate industry standards to ensure compliance and identify necessary security controls for the application.
Integrate Security Solutions for Applications	40	Put the developed security solutions into both Android and IOS applications, ensuring secure user authentication and data handling.
Integrate Security Solutions into Database	40	Implement encryption and secure access protocols within the database to protect sensitive data at rest and during transactions.
Penetration Testing	30	Conducted comprehensive penetration tests on both applications and the database to identify vulnerabilities and strengthen security measures. This includes testing the original application to find security gaps and testing the new iOS and Android applications to ensure the gaps were covered.

3.7 Other Resource Requirements

Resources needed to complete this project are the physical weight rack and sensor. These both already exist and will be utilized to connect to our application. Another resource required to complete this project is the database. TrueForce Technologies has an existing database that our app will connect to, in order to save specific information for each user.

Appendix

3.2.1 Task Decomposition Flowchart



3.4.1 Gantt Chart