COMP6000 Technical Report

PT Manager

Billy Spiers, Daniel Nkurunziza, Hisham Javed, Kevin Hills

Supervised by: David Castro-Perez (d.castro-perez@kent.ac.uk)



School of Computing

University of Kent

**Abstract**

Project description

An app for managing schedules and classes for personal trainers and clients, to book for classes and communicate with each other. The app uses a database to store client and trainer data and messages to be used for better management and communication between them.

Results

The PTManager app permits both clients and trainers to login and gain access to specific features to manage their schedules easily. The client can log in at any time if they have registered and view their workout classes/programs using a database calendar. Trainers can alter their sessions as well as create ongoing workout programs that span several days or weeks. With a class/program created, the clients and trainers can communicate in a group chat room made for the class/program for updates or progress.

**1. Introduction**

This project aimed to address the logistical challenges faced by both personal trainers and clients by providing a comprehensive platform for managing schedules, booking classes, and facilitating communication.

The PTManager app is designed to streamline the scheduling process for both trainers and clients, allowing them to easily coordinate appointments and manage their respective agendas. By harnessing the power of a database, the app securely stores client and trainer data, as well as messages exchanged between them. This not only enhances organization but also fosters improved communication between trainers and their clientele.

Our efforts have led to the development of a versatile application designed to benefit both clients and trainers. With this app, clients can conveniently log in to access personalized schedules tailored to their fitness goals. The app seamlessly integrates with a database, allowing clients to view upcoming workout classes or programs through a user-friendly calendar interface.

Moreover, the app facilitates real-time communication between clients and trainers through dedicated group chat rooms tailored to specific classes or programs. This feature enables timely updates, progress tracking, and fosters a supportive community atmosphere conducive to achieving fitness goals.

In this technical report, we delve into the key parts of the PTManager app, exploring its functionalities, design considerations, implementation process, and the implications of our findings. Through rigorous testing and analysis, we aim to provide insights for further refinement and optimization, ultimately enhancing the user experience and effectiveness of the app in meeting the needs of personal trainers and their clients.

**2. Background**

**3. Aims**

The primary aim of the PTManager project is to develop a comprehensive application that addresses the scheduling and communication needs of both personal trainers and their clients within the fitness industry. This includes:

Streamlining Scheduling Processes: The app aims to simplify the scheduling of training sessions and classes for both clients and trainers. By providing an intuitive interface and seamless integration with a database, users can efficiently manage their schedules, book classes, and make necessary adjustments as needed.

Enhancing Communication Channels: Effective communication is essential for fostering strong client-trainer relationships and ensuring clarity regarding session details and progress tracking. The PTManager app seeks to facilitate seamless communication between clients and trainers through features such as real-time messaging and group chat rooms dedicated to specific classes or programs.

Improving User Experience: Central to the project's objectives is the creation of a user-friendly interface that caters to the diverse needs and preferences of both clients and trainers. By prioritizing ease of use and intuitive navigation, the app aims to enhance the overall user experience, making it accessible and enjoyable for individuals at all fitness levels.

**4. Project Requirements**

The focus of our project was to facilitate the access a personal trainer and client have to their programs, classes direct communication with each other. Providing the fundamentals both sides would need when registering in the app; signing up for classes, taking part in ongoing programs, being able to alter or update classes/programs and communicating with other users.

**4.1 Trainer/Client**

At the beginning of the project, we discussed user requirements; settling on the idea that the app should be split into two separated sides, client side and trainer side, where each side contains unique functionalities.

The client user would be any user that uses the app as a client/customer, who is looking for fitness activities to take part in. They should be able to log in to see and interact with activities/programs posted by trainers. Trainer users would be professional trainers using the app to post and manage the fitness activities they personally run. They should be able to log in to manage their own activities and create more.

Users should only be able to see functions within their own scope. For example, clients should not be able to create activities, and trainers should only be able to edit activities they posted themselves.

**5. Software Design**

**5.1 Initial design**

As a team, we decided to kickstart our PTManager app project by creating initial designs using Android Studio. This choice allowed us to get comfortable with the software while laying the groundwork for our app's development. Our aim was twofold: firstly, to provide a platform for giving and receiving feedback on our designs. By critiquing each other's work, we could identify areas for improvement and refine our ideas collaboratively. Secondly, these initial designs helped us visualize our app's interface and features, guiding our development process. Through open communication and iterative refinement, we aimed to create a user-friendly and functional app that meets the needs of both personal trainers and clients in the fitness industry. By immersing ourselves in the design phase early on, we ensured that our team had a clear understanding of the app's objectives and user requirements, setting a solid foundation for the subsequent stages of development. Additionally, creating these initial designs allowed us to identify any potential challenges or limitations early on, enabling us to address them proactively and minimize delays in the development process. Overall, this collaborative approach to design not only facilitated the creation of a more polished and user-centric app but also fostered a sense of teamwork and shared ownership among team members.

**5.2 Final Detailed Design**

The team were keen to experience developing mobile applications using Android Studio and Java programming language for the front-end GUI side.

None of the team had used this development framework before so training time was factored into the project plan/milestones before development properly began.

Each team member became responsible for designing the frontend and backend for their own app features to equally spread the workload and knowledge.

One of the initial tasks was for the team to create screen and database prototypes based on the user stories/requirements and research into other similar apps. We all voted on which design was the best fit for the project based on factors such as complexity and professionalism.

For this project the team decided it was more important to focus on how the app works and communicates with the backend database.

For the backend side, we decided to use industry standard software consisting of a MySQL database (which we were all familiar with), plus PHP server software along with PHP script files to communicate with the database.

The PHP script files were named based on the app feature they relate to. This makes it easier for a developer to find and debug where necessary. The files include a check for the existence of session data. If session data is not found, then the message sent back to the Android app causes the app to close all screens and restart the login process.

A standard HTTP URL connection library and methods (within Android Studio) were used to send and receive requests to/from the PHP server.

A generic java class was created (“ServerConnection”) which allows any activity to make a connection to the PHP server and pass data between the database and the app.

Each app feature has one or more backend PHP script files containing SQL queries specific to that feature's data processing needs.

To make things as easy as possible for this project, the "ServerConnection" code and PHP script files use the default HTTP "GET" request method for all types of database queries.

A future enhancement would be to make the PHP script files more industry standard by having separate files for each type of SQL query (e.g. SELECT/UPDATE/DELETE) along with the appropriate database request method (e.g. GET/POST methods for SELECT type queries, PUT for Insert type queries and DELETE for delete type queries).

To try and prevent SQL injection attacks, PHP files use “MySQLi” and “PDO” prepared statements. Any passed arguments are also checked for validity.

The server connection is performed asynchronously (using the standard “AsyncTask” background method) to ensure the app continues processing whilst the data is being sent/retrieved.

The “AsyncTask” method handles the response from the PHP server and returns the data back to the appropriate activity (using the “ProcessFinish” method).

Due to some activities requiring multiple database calls, a variable was added to the server connection (“Destination”) to identify what the data refers to, so the activity knows what actions to take with this data.

As a potential future enhancement, the data identifier could be placed within the data being returned from the PHP server.

The "ServerConnection" class connects to the PHP server using a standard format web URL consisting of the PHP server IP address and port along with the PHP script name and arguments relating to the SQL query to be performed against the database.

The connection is opened and closed each time for every database request. Due to possible bottlenecks and the likelihood of delays/errors when a widely used app performs lots of connections and disconnections, a future enhancement would be to create one connection and keep it open until the app is closed.

The data returned from the PHP server is in a standard JSON encoded format and contains a status code indicating whether the request was successful or not.

Due to there being a lot of source files and layouts in Android Studio, a naming standard was implemented to make it as easy as possible to see which files belong to each feature and to avoid possible duplicated class names etc.

Each java and layout file contains the name (or shortened version) of the feature it belongs to.

**5.3 Language and Environment**

**Database environment:**

**Language environment:**

**6. Implementation of project**

**Android Studio**

We developed our project with a combination of three technologies, Android studio for front-end development, MySQL for the database management and PHP for the server-side scripting. We created java projects as prototypes to familiarise ourselves with the IDE as a starting point implementing the list of features each with placeholder values. From this we would build our interface design and pick out key elements crucial to displaying data from the database efficiently.

The main features included new user registration, calendar search, programs/classes search, client to trainer chat and a notification system.

**Backend connection**

For the backend development, we created PHP scripts to handle numerous functionalities such as data retrieval, client/trainer login authentication and program updates among other features.

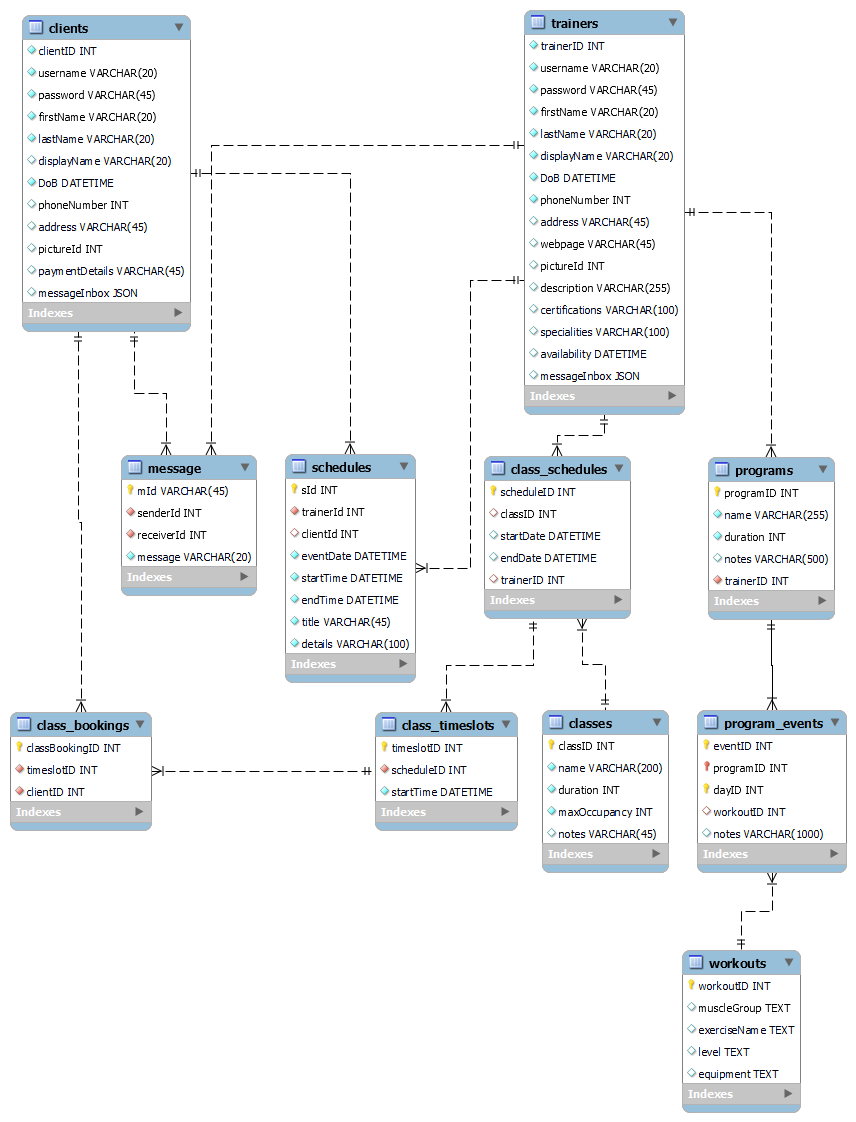
**6.1 Database Design**

The user stories/requirements, along with research into other existing applications, formed the main basis for the database design. An initial database design diagram was created detailing the main base set of tables. The user type (Clients and Trainers) is the main starting point for the database design. For this project, the two user types have their own tables (clients and trainers) and use an ID type primary key to link to all the other tables. The team did consider another approach using a “User” table along with a “Roles” table.

The initial base set of tables included clients, tables, message, schedules (for the calendar).

Tables related to the programs and classes features were later added once development had started on those features. See the diagram below for further information.

The team also created a single import file (“appdb.sql”) containing all table creation queries to make it easy for anyone else to setup the database. This file also includes some test data for app demonstration purposes.



**6.2 New user registration screen**

**6.3 Log in screen**

**Design**

As mentioned previously, we designed this app around two distinct user types: clients (the customer and average user) and professional trainers, where each user will have access to different features. Therefore, we required authorisation between the two user account types, accomplishing this by implementing two separate login screens.

When developing this, we made two assumptions about the userbase: that client users will outnumber trainer users, and that professional trainers should be more acquainted with the app's interface and features via frequent use and training. Using these, we decided to set the client login screen to be the default login, with an option to switch to a trainer login screen. This provides a clean and user-friendly experience for the average user by separating them from the professional features which clients should not be interacting with.

We also had the option to keep the login as one combined function that will check the account type after the user inputs their details, however we decided to keep them separate to allow a user to use their details for both account types; e.g. a trainer can sign up as a client with the same information. When developing the database, this also seemed like the correct choice since the data fields for each account type varied a lot.

**Security**

After the user enters their details and the "Log In" button is pressed, the password is encrypted using MD5 before the details are sent over HTTP to the backend to be processed. The password is encrypted to ensure this sensitive data is secured, both during transmission and storage.

While this works as basic security, if we had more time to work on this project, then this would be developed for better security. Ideally, the type of encryption being used should be more resistant, such as SHA-256, and HTTPS should be used instead of HTTP to increase security during transmission.

**Backend and Session Data**

After the details are sent to the server upon "Log In" button press, checks are executed to see if an account exists within the database containing the same username and password, sending back a response either confirming or denying. If the details are correct, the server will save them as a session, which is necessary for maintaining the user's authenticated state throughout their use of the app's different features as well as allowing the app to personalise data. This session data is contained and manipulated only within server, and requires an updated session check each time the app requests the data.

**6.4 Client menu screen**

**Design**

With most of the features being accessible for the client and trainer we needed a user-friendly interface that would allow the logged in client or trainer to use the features designed for them. We decided to create a homepage/menu that would link all the features after the user logs in.

Based on our research in UI design, icons are the best option for quick visual communication for the user to understand the functionality of the different elements available on the screen. Due to the limited space available on mobile screens text labels would not be compact enough to match the features shown and well-designed icons can be universally understood across various cultures and languages.

**Navigation**

Once a user has logged in with the correct credentials, they are directed to the homepage that contains tiles of icons that represent the different functionalities of the PT app. At the top of the screen a navigation bar was implemented for each page displayed in the app aside from the log in page. Set with recognisable icons for that will give them access to the homepage if the user is goes to a feature and their notifications for new information.

**6.5 Trainer menu screen**

While the Trainer Menu shares some similarities with the Client Menu in terms of user interface and navigation structure, it offers distinct features and functionalities tailored to the specific needs and responsibilities of trainers. Such features include program creation, class creation and alteration that can only be accessed by the user with the correct credentials for a trainer login.

**6.6 profile search/display screen**

**6.7 Calendar search/display screen**

The Calendar page in our app was carefully crafted to simplify the process of scheduling workouts and classes for both trainers and clients. First off, we created a Calendar Search screen, allowing trainers to easily check their clients' schedules and vice versa. This feature helps trainers plan sessions efficiently and communicate effectively with clients about their availability.

Next, we developed the Calendar display screen, acting as a central hub where all booked classes and events are showcased. Clients can quickly glance at their upcoming workouts or classes, while trainers can organize their schedules and keep track of client progress. We made sure the interface is easy to navigate, with simple clicks to explore dates and event details.

Additionally, we added a filter option to the Calendar page, making it even simpler to manage appointments and events. This filter tool lets trainers and clients easily find specific entries on the calendar, making editing or deleting items a breeze.

With the filter option, users can narrow down their view of the calendar based on things like date ranges, types of events, or specific client or trainer schedules. For example, trainers can quickly find sessions with clients, and clients can easily locate their own scheduled workouts.

This filter feature gives users more control over their calendar, allowing them to navigate through their schedules quickly and efficiently. It's another way we've made our app user-friendly and adaptable to the needs of both trainers and clients.

The messaging page underwent thorough testing to ensure its functionality and reliability. We employed a testing approach that involved adding dummy contacts and dummy messages to simulate real-world usage scenarios. This allowed us to assess various aspects of the messaging page, including message delivery, synchronization across devices, and the performance of real-time updates

**6.8 Programs feature screens**

This feature consists of several screens related to workout/exercise programs.

The feature is accessed via the “Home Page” menu.

A PHP script file (“programs.php”) contains functions for all the database queries that this feature uses.

**Programs List screen**

This is the screen first shown. At the top of the screen there are fields to allow various optional search criteria to be chosen.

At the bottom is a list of programs that match the search criteria populated using data from the “programs” table in the database. All programs are listed by default.

The name search field allows a user to enter partial or full words/digits. The minimum and maximum day fields allow digits to be entered to represent the desired duration of the program.

There is also a dropdown list/spinner allowing a particular trainer to be chosen. This list is populated using data from the "trainers" table in the database.

Once search criteria have been chosen the "Search" button can be selected which updates the list of programs accordingly.

If a program is selected from the list, then the "Program Create/Edit" screen is shown to allow the program information to be viewed or edited.

There is also a "Create" button that is only enabled for trainers so that they can create a new program. This button also shows the "Program Create/Edit" screen.

To exit the screen the user selects the default phone "Back" button which will return to the "Home Page" menu.

**Program Create/Edit Screen**

This screen is used in a multi-purpose way to view, create, edit and delete programs.

The screen shows the program data fields plus buttons at the bottom to cancel, delete, save, show the “Day Planner”.

If an existing program was previously selected, then the screen is populated with data from the chosen program. If the user is the trainer that created the program, then the program data can be edited. When selecting the “Save” button, the user must confirm they wish to update the program via a dialog message that gives a warning that changing the duration could cause workout/exercise event data loss. Upon confirmation, any existing workout/exercise event data for days greater than the duration are deleted from the “program\_events” table in the database along with updating the “programs” table.

Otherwise, the screen data is set as "read only". In this case, the delete and save buttons are disabled.

If the screen was opened using the program list "Create" option, then all program data fields are empty and can be entered. The "Save" button will be enabled to allow the new program to be saved in the database.

The “Cancel” button will return the user to the “Program List” screen. The “Delete” button will remove the program from the programs table in the database.

The “Day Planner” button shows the “Day Planner” screen which lists all workout/exercise events for the duration of the program.

**Day Planner Screen**

This screen lists all workout/exercise events for each day of the program.

Depending on the duration of the program, the screen is built dynamically to show each day followed by any events for that day.

Each listed day has a button to add events which is only enabled if the user is the trainer that created the program. The “Add” button shows the “Program Event Create/Edit” screen to allow event information to be created.

If an event is selected from the list, then the "Program Event Create/Edit" screen is shown.

**Program Event Create/Edit Screen**

This screen is used in a multi-purpose way to view, create, edit and delete program events. It shows the event data fields plus buttons at the bottom to cancel, delete or save.

If an existing event was previously selected, then the screen is populated with data from the chosen event. If the user is the trainer that created the program, then the event data can be edited. Otherwise, the screen data is set as "read only". In this case, the delete and save buttons are disabled.

If the screen was opened using the day planner "Add" option, then all program data fields are empty and can be entered. The "Save" button will be enabled to allow the new program to be saved in the database.

The “Cancel” button will return the user to the “Day Planner” screen. The “Delete” button will remove the event from the “program\_events” table in the database.

**6.9 Classes feature screens**

This feature consists of several screens related to creating and booking fitness classes.

The feature is accessed via the “Home Page” menu.

A PHP script file (“classes.php”) contains functions for all the database queries that this feature uses.

**Classes screen**

This is the screen first shown. At the top of the screen there are buttons to access the “Class Bookings” and “Class Schedules” screens. The “Class Bookings” screen allows a client to view and cancel any of their timeslot bookings. The “Class Schedules” screen allows a trainer to view and cancel any of their scheduled timeslots. These buttons are enabled/disabled depending on whether the user is a client or trainer.

There are also fields to allow various optional search criteria to be chosen.

At the bottom is a list of classes that match the search criteria populated using data retrieved from the database. The classes shown are ones that have a future date time slot.

The name search field allows a user to enter partial or full words/digits. The minimum and maximum duration fields allow digits to be entered to represent the desired duration of the class.

There is also a dropdown list/spinner allowing a particular trainer to be chosen. This list is populated using data from the "trainers" table in the database.

Once search criteria have been chosen the "Search" button can be selected which updates the list of classes accordingly.

If a class is selected from the list, then the "Class Info” screen is shown to allow the class information to be viewed.

There is also a "Create" button that is only enabled for trainers so that they can create a new class schedule. This button shows the "Class Schedule" screen.

To exit the screen the user selects the default phone "Back" button which will return to the "Home Page" menu.

**Class Bookings Screen**

This screen shows a list of all future bookings that a client has made. To the right sude of each booking is a “cancel” button which allows that booking to be cancelled. A dialog message asks the client to confirm they wish to cancel the booking at which point the client is removed from the “class\_bookings” table in the database.

To exit the screen the user selects the default phone "Back" button which will return to the "Classes" screen.

**Class Schedules Screen**

This screen shows a list of all future scheduled timeslots that a trainer has made. To the right side of each timeslot is a “cancel” button which allows that timeslot to be cancelled. A dialog message asks the trainer to confirm they wish to cancel the timeslot at which point the timeslot is removed from the “class\_timeslots” table in the database. Any bookings related to this timeslot are also removed from the “class\_bookings” table.

To exit the screen the user selects the default phone "Back" button which will return to the "Classes" screen.

**Class Info Screen**

This screen shows all data related to the class previously selected.

The screen shows the program data fields which are “read only” plus buttons to cancel and book. The “Cancel” button will return the user to the “Classes” screen. The “Book” button is enabled only for clients and shows the “Class Timeslot Book” screen to allow the user to book a timeslot for this class.

**Class Timeslot Book Screen**

This screen has a dropdown list/spinner containing all future timeslots available for the selected class. There are buttons to cancel and book.

The “Cancel” button will return the user to the “Class Info” screen. The user chooses a timeslot from the dropdown list and then clicks the “Book” button at which point the “class\_bookings” table is updated in the database.

**Class Schedule Screen**

This screen has

**7 Chat search/display screen**

The messaging page in our app was carefully crafted to make communication between trainers and clients smooth and intuitive. We designed two main screens for this: the Chat display screen and the Chat search screen.

The Chat display screen is where users can see all their ongoing conversations. It shows who's in the chat and lists all the messages exchanged. You can easily send new messages from here, keeping the conversation flowing effortlessly.

On the other hand, the Chat search screen helps users find specific chats quickly. It's like a filter tool. By default, it shows chats the user is already part of, making it easy to jump back into ongoing conversations.

Both screens are connected to our database, ensuring that all chat-related information is securely stored and readily accessible. This means messages, participants, and conversation history are always available whenever users interact with the messaging page. Additionally, robust security measures are in place to protect the confidentiality of conversations and prevent unauthorized access.

Overall, our messaging page offers a simple and reliable way for trainers and clients to stay connected, fostering effective communication within our fitness community.

**7.1 Notifications display screen**

**Design**

For this feature, both client and trainer would need access to any new information such as changes or updates to programs and classes. We do not need different layouts for the client and trainer, we just need to pick different data from the appDB table in the database.

Since the function is picking recently added classes/programs in the database it generates the most recent data according to timestamps in the table and after being viewed the data is deleted with the 'refresh’ button. This applies to both client and user since both parties will have access to the most recent data, especially the trainer that would be notified when a client has signed up for their class/program.

**PHP**

**7.2 Testing**

**PHP**

For testing PHP functions, we used two methods: manual testing through a browser using echoed messages for debugging or developing dedicated PHPUnit tests.

Manual testing was straightforward and accessible for quick checks and debugging, however it had potential to get messy with the output, and the coverage was up to the developer's knowledge of the system.

PHPUnit allowed for deeper testing with wider coverage of inputs/outputs as well as easy automation, meaning it would likely be more efficient if the backend was more complex. Unfortunately, PHPUnit was new to all of us and took up a lot of time to set up and learn. Eventually it was pushed aside so we could focus on higher priorites and if we had more time for this project, PHPUnit tests would be developed for all server-side processes.

**Database**

When testing with the database we set up test tables to store test data, which was often manually inserted. These tables mimicked our full table designs, although sometimes started off with less columns, prioritising only necessary columns for the function being developed. This allowed us to test any interactions between the backend and the database, ensuring the intended functionality of SQL queries.

Because of how the app is structured, many of our functions are based on parsing or manipulating user data, therefore setting up working and accurate test tables was crucial for the development process.

**8. Conclusion**

**9. Future Work**

**Acknowledgements**

**Bibliography**