[Techniques Used 44](#_Toc511631510)

[Coding Styles 44](#_Toc511631511)

[Group A Techniques Used – Model 46](#_Toc511631512)

[Group A Techniques Used – Algorithm 47](#_Toc511631513)

# Techniques Used

## Coding Styles

Coding Styles are highlighted based on the level of difficulty

|  |  |  |
| --- | --- | --- |
| Style | Evidence | Coding Page No |
| Basic | Meaningful identifier names | Page 88, Page 90, Page 109, Page 115 |
| Annotation used effectively when required | Page 102, Page 114, Page 116, Page 91 |
| Good | Well Designed User Interface | Page 133, 131, 135, 128, 125 |
| Modularisation of code | * Split into Classes: User, Routine, Exercise, Timetable, Search, EmailObj |
| Minimal use of global variables | Page 90 |
| Managed casting of types | Page 110, 99 |
| Use of constants | Page 98 |
| Appropriate indentation | Check code of any class or page |
| Self-documenting code | 97, 100, 117-118, |
| Consistent style throughout | Camel Case for variables and functions (Check meaningful identifier names, same page numbers) |
| File path parametrised | Page 135, 89 |
| Excellent | Modules (subroutine) with appropriate interface | Aggregation:  Page 90: Exercise is declared in Routine Class  Page 105: Routine is declared in Search Class  Page 110: Routine declared in Timetable Class  Use of methods & properties  Page 93: Exercise’s class public methods are called within the Routine Class  Page 103-104: Routine’s class public methods are called within the Search Class ~ The search class performs a sort on several routine objects.  Routine objects are associatively aggregated in Search Class ~ Used to display routines from the result of a search |
| Loosely coupled modules (subroutines) – module code interacts with other parts of the program though its interface only |
| Cohesive modules (subroutines) – module code does just one thing | Example of a method from each class  *Class Name: Purpose of class*  *Example of method in class*  Search Class: Searches for Routines  Page: 102  Timetable Class: Generates Timetable  and adds/remove routine  Page: 106  Exercise: Holds information about each exercise  Page: 86 |
| Defensive programming | 1. Disable form, once data has been entered, preventing double entry. Page 94 2. Double Password Check: a) Creating Profile b) Changing Password: Pg 125 3. Presence Check: Pg 125 4. BindValue, prevent SQL injections: Page: 85, 89-90, 95 5. Correct format for inputs e.g Integers for: Reps, Sets, Weight Entry, Dropdown list for limited inputs e.g Name of Routine: Pg 116-117 |
| Good exception handling | 1. Wrong Entry Log In: Pg 112-113 2. Incorrect double entry of password: Pg 127 3. Incorrect upload of photo: Pg 125 4. Trying to add routine to a day which already has a routine: Pg 106 5. Trying to remove a routine from a day without routine: Pg 107 6. Email Cant be sent: Pg 131 |

## Group A Techniques Used – Model

|  |  |  |
| --- | --- | --- |
| Task | Explanation | Evidence |
| Complex data model in database | Check Entity Relationship Diagram Model to see the structure of the database. | *Design – Page 5*  Log of Evidence - Fig 3.1.0 |
| File Organised for direct access | Profile Pictures stored in 1 folder  Pictures for exercises stored in 1 folder | *Log of Evidence -* ***Fig 2.4.1*** |
| Complex user-defined use of object oriented programming (OOP) model e.g classes, inheritance, composition, polymorphism, interfaces | Composition  Exercise is only instantiated in the Routine Class, and cannot exist without it  Association  Routine class is instantiated associatively in Timetable Class; when a user wants to add a routine to their timetable and in the Search Class; when a user simply wants to view the routine. | **Aggregation:**  Page 90: Exercise is declared in Routine Class (Composition)  Page 105: Routine is declared in Search Class (Association)  Page 110: Routine declared in Timetable Class (Composition)  **Use of methods & properties**  Page 93: Exercise’s class public methods are called within the Routine Class  Page 103-104: Routine’s class public methods are called within the Search Class ~ The search class performs a sort on several routine objects. |
| Cross table parametrised SQL | Using one sql statement to search through multiple data, using a common “link”. | *Log of Evidence -* ***Fig 2.5.2*** |
| Dynamic generation of objects OOP model | **User**   * An instance of user is created when the user creates a profile, registers or logs in. (Page 121)   **Routine**   * The routine stores an array of exercise classes. Exercise Class is instantiated only in the routine class. (Page 90) * The users access properties from both types of classes when performing a routine. * There are 2 types of ways a user can *view* a Routine, one is through the Search Class (Page 105) and the other is through Time Table Class. Hence the routine is associatively instantiated in these 2 classes. * When the user performs a routine, the user must have first added it to their timetable, the system produces a template with feedback boxes and suggested weight. * When the user views the routine through the Search class, the routine is viewed as a template and does not contain feedback boxes or suggested weight.   **Timetable**   * An instance of timetable is created when the user goes onto the timetable page. * Timetable has associated properties and methods relating to the storing of dates and associated routines, which the user inputs.   **Search**   * An instance of search is created when the user goes onto the search page. The search class helps sort the results and store them. (Page 129) | **Aggregation:**  Page 122: User object is instantiated when logged in  Page 90: Exercise is declared in Routine Class  Page 105: Routine is declared in Search Class  Page 110: Routine declared in Timetable Class |

## 

## Group A Techniques Used – Algorithm

|  |  |  |
| --- | --- | --- |
| Group A Technique Used | Explanation | Evidence  Appendix Page Number |
| Cross table parametrised SQL | In Exercise Class: Finds all the exercises with which trains a specific muscle group. | 89 |
| Aggregate SQL functions | In Routine Class: Finds average rating of a specific routine using all individual ratings given by users.  In Routine Class: Finds the number of exercises in a routine by counting the number of exercise ids | 91 |
| Complex user-defined algorithms | **Dynamic Feedback System:**  When User has performed a routine before (in the last 31 days): the system looks at their most recent feedback for each exercise in the routine. It then prints a suggested weight or range of weights. This system does it so that the “Challenging” option takes priority, as the “Challenging” level is what users ought to be working on.  *It takes the highest challenging weight out of all the sets of an exercise*  *If not it takes the highest “Easy” and the Lowest “Too Difficult” to create a range e.g Greater than 20kg, Lower than 30kg*  It also has an immediate feedback system that is simply a suggestion for users: For exercises that have a rep range 6 or below, the system gives a suggested weight of +5kg (if too easy) or -2.5kg (if too difficult). For rep ranges greater than 6, the system gives a suggested weight of +2.5kg(if too easy) and -2.5kg(if too difficult).   * **Input Feedback and Weight**: To ensure that each feedback in the system can be *uniquely* identified, each feedback input is unique and is created by the combination of the order number of the exercise, either f or w (feedback or weight) and the exercise id. These 3 pieces of data concatenated together will allow the system to uniquely identify, store and use the data. (Page 88) * After the user performs the routine, the system will store the rating given by the user. This along with ratings given by other user, will form the average rating of the routine which will affect the position the routine appears in when it is searched for. * A log is also used to show the when routines are performed and the ratings. | 101 – 105  Functions  displayRoutine()   * generateSuggestedWeight   Pg 98   * implementFeedback()   Pg 95 - 99 |
| Sort | **Search**   * ­The User can search for a routine based on their Name, Creator or Muscle, and rank it in order of rating or date performed. A bubble sort is used to rank the results of a search.   The search is done so that the highest rating is first, or most recently performed is first. | **Page 102-103** |