Redo use case templates to be based upon use cases from:

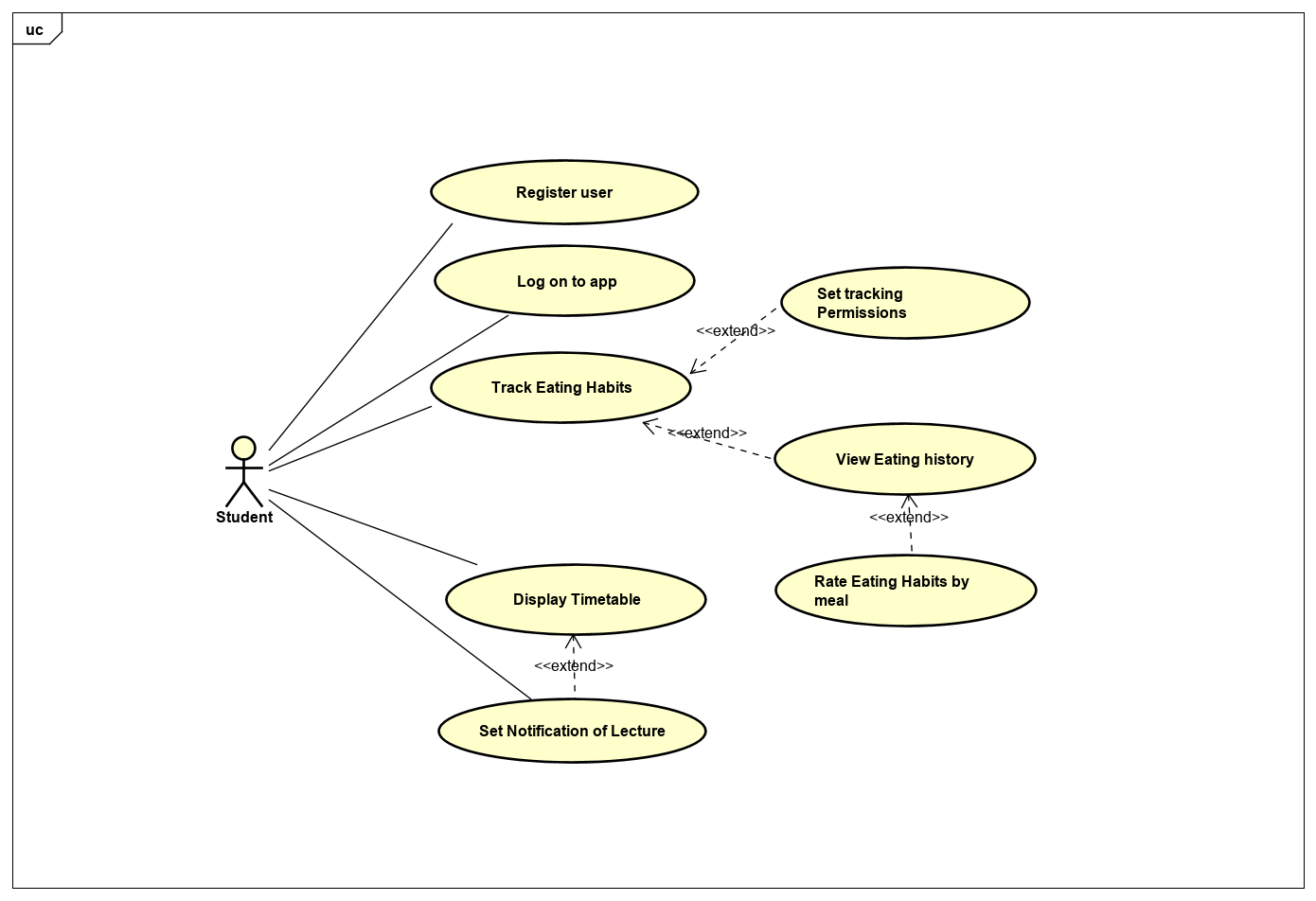
Register user is the one we already have as student sign up.

Still need to do “log on to app”, “track eating habits” with sub templates “set tracking permissions”, “view eating history”

“Display Timetable”

Link to effective pair programming

<https://gds.blog.gov.uk/2018/02/06/how-to-pair-program-effectively-in-6-steps/>



Prioritise and MoSCoW ordering

**Tasks**

1. Review your group’s part 1 of the assignment, including mockups and identify where more clarity or detail might be needed. Revisit the **use case templates** and make sure you’ve documented them in enough detail so that you are clear what must happen in each case, and what data items are involved. Copies of any relevant forms, documents etc. can help here
2. Prioritise **user stories** and order the for implementation (MSCW), seeing if you can identify any *dependencies* between these. Which use cases might these correspond to?
3. Identify the main **entity classes** and draw a class diagram, showing each classes attributes and the relationships between classes. Relationships should be labelled showing their cardinality. If any other classes are needed (e.g. controller, manager classes) add these also.
4. Draw at least *one* **state machine diagram** for a class where state change is important to consider. Use this to ensure that you have enough methods to ensure that state changes can be achieved.
5. For three methods that are most important to the features you wish to implement in code, write **method specifications**.
6. Write **code** to implement your chosen user stories/use cases, using the classes you have identified. Use paired programming to help ensure that code is clear, readable and well structured.

|  |  |  |
| --- | --- | --- |
| **USE CASE 1** | Track Eating Habits | |
| **Goal in Context** | To allow the user to track their eating habits and set/access permissions for tracking of their habits. | |
| **Scope & Level** | Primary User Task | |
| **Preconditions** | User is registered on app and has either credit/debit card/loyalty card at O’Hehirs | |
| **Success End Condition** | User can view tab with record of eating habits display details stored | |
| **Failed End Condition** | Student not able to display eating history or set permissions for tracking | |
| **Primary,**  **Secondary Actors** | User is primary, secondary actors are O’Hehirs sales DB system and user tracking system | |
| **Trigger** | User is already logged on, selects Habit Tracking tab in app | |
| **DESCRIPTION** | **Step** | **Action** |
|  | 1 | User selects tracking tab |
|  | 2 | Tracking Data overview is displayed on screen |
|  | 3 | Set permissions button is displayed at bottom/top right of screen |
|  | 4 | User has option to show details of day/week/months eating habits. Requires an int variable for display period |
|  | 5 | Results display on screen according to user selection (preferred display type should be saved as a variable ) |
|  | 6 |  |
|  | 7 |  |
| **EXTENSIONS** | **Step** | **Branching Action** |
|  | 3a | User can select permissions and then set tracking type/permissions. Go to separate use case template for permission setting |
|  |  |  |
|  |  |  |
|  |  |  |
| **SUB-VARIATIONS** |  | **Branching Action** |
|  |  |  |
| **OPEN ISSUES** |  |  |

Variables used by tracking habits user case include:

int for displayPeriod (1= day, 2= week, 3= month),

int for trackingMethod selected eg: 1 = Loyalty Card, 2 = CC, 3 = Debit card

//Protein level for foods over selected period

//Fat Level for foods over selected period

Food purchases added to list of food objects which have date, food scores for protein/carbs/fat etc

User class will hold details of user, track their permissions and maintain a list for food objects consumed by a user

Variables: name, student number, course, permissions for tracking,

Methods: to add user details; add food purchases to a list (which will assign a date/time to each purchase); retrieve timetable info based upon user course variable;

string name, studentNumber, course,

int tracking permissions (maybe use a similar system to rwx: 4, 2, 1 notation)

Foods class will hold a name for each food object, give ratings for each food object

Variables for protein levels, fat levels, carb levels, food name, overall health rating

Methods: to add new food items; rate food items

|  |  |  |
| --- | --- | --- |
| **USE CASE 2** | Setting permissions for habit tracking | |
| **Goal in Context** | To allow the user to set tracking permissions based upon card type/loyalty card. | |
| **Scope & Level** | Primary User Task | |
| **Preconditions** | User is registered on app and has one or more of credit/debit card/loyalty card at O’Hehirs | |
| **Success End Condition** | Permissions are set for tracking types used | |
| **Failed End Condition** | Student not able to set permissions for tracking | |
| **Primary,**  **Secondary Actors** | User is primary, secondary actors are O’Hehirs sales DB system and our own user tracking system | |
| **Trigger** | From Habit Tracking tab in app user selects set permissions “cog” | |
| **DESCRIPTION** | **Step** | **Action** |
|  | 1 | User selects set permissions |
|  | 2 | User adds/removes permission tick for chosen tracking type eg: O’Hehirs Loyalty card/CC/Debit card |
|  | 3 | Bools for tracking variables are updated at app level/tracking system db on confirmation of changes. |
|  | 4 | App window reverts back to habit tracking page |
|  | 5 |  |
|  | 6 |  |
|  | 7 |  |
| **EXTENSIONS** | **Step** | **Branching Action** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| **SUB-VARIATIONS** |  | **Branching Action** |
|  |  |  |
| **OPEN ISSUES** |  |  |

User properties

enterUserName(string name)

* Name = name;

enterStudentNumber(string stdNumber)

* studentNumber = stdNumber;

enterCourseCode(string code)

* courseCode = code;

trackingPermissions : int must be set to zero on user creation

* see setTrackingPermissions() method below for setting permissions

userHealth : int

Method specs

displayRatings()

Takes variables for displayPeriod and userHealth and run through a switch statement that works by comparing user intake for the day/week/month vs an acceptable limit

setTrackingPermissions()

sets int for permissions using 4,2, 1 system similar to R,W,X file system permissions.

If statement adds the corresponding number if a permission is granted Loyalty card = +4, CC = +2, Debit = +1. So trackingPermissions must be set to zero when a user is created and can then change to 0-7 according to permissions set.

addNewFood()

takes (name : string, protein : int, carbs : int, fat : int, overallHealthRating : int) and outputs as a Food object which is then added to the <Foods> list

usersPurchasesList()

initialises a List <Foods purchased> for each user

addPurchases ()

takes a Food object adds a DateTime for the purchase and adds the resulting object to the users List <Foods purchased>

displayRatings()

Takes variables for displayPeriod and userHealth and run through a switch statement that works by comparing user intake for the day/week/month vs an acceptable limit

Food Purchases Manager methods:

CompareTo(object obj) //will allow for entries in FoodsPurchased list to be sorted according to purchase date

{

//get reference to next object in list

Food objectToCompareTo = obj as Food;

//compare to is set to compare the Purchase Date

int returnValue = this.datePurchase.CompareTo(objectToCompareTo.datePurchase);

return returnValue;

}