1809ICT Assignment 1

Project Requirement Analysis and Specification

Student 1:

Name: Scott **McMullan**

Student ID: **s2916255**

Date and time of laboratory: **Thursday, 1600**

Student 2:

Name: **Leon Andov**

Student ID: **s5088455**

Date and time of laboratory: **Friday, 1200**

Student 3:

Name: **Joshua Baker**

Student ID: **s5057501**

Date and time of laboratory: **Friday, 1400**

Section 1 – Set Up a Project

Name of The Proposed Software Project

Health RPG

System Vision Document

System Vision

A client, who is a health and fitness enthusiast, is striving to create an app that allows users to easily construct diet plans with a 'gamer' twist. There are numerous apps available for purchase that do not meet the requirements or interest of mainstream users on mobile devices. Hence a health app which has an outstanding, yet reliable system will have an exceptional performance on the digital market.

The practicality of a nutrient-tracking app is to provide an effective solution for people who are concerned with maintaining a healthy body. The application will be a simple system where users can setup an account, monitor their progress towards personal goals and ultimately improve their lifestyle through consistently observing their consumption.

However, the client wishes to implement a 'gamer-twist' concept of levels and achievement badges which virtually reflects a user's activity. Essentially, a level system where users can earn experience points by completing tasks aligned to their personal diet will be embedded into the application, as well as the ability to earn virtual badges which represent respective achievements. It is proposed that, this along with the diet plan will create sustaining user activity and competition with other health apps.

Therefore, to take advantage of the mobile app market, rolling out an authentic health app will provide a sufficient diet-management product. It is proposed that people will have the ability to download a free application that contains advertisements and a paid application that has no advertisements.

Software System Capabilities

The following list outlines the proposed capabilities for the application.

- Users will have the ability to create a secure account.
- Users will have validated achievement badges tied to their account which will display their accomplishments.
- A unique, yet simple layout which can be effortlessly navigated.
- Users can create and customize their own personal diet plan.
- Users will have access to a 3rd party database using the application API which will allow them to calculate their daily nutrient intake.
- Users are able to track their progress towards a personalized objective: i.e. strength, endurance, weight loss/gain.
- The app accommodates users with different body types and will supply them with the option of diet templates.
- Users will have access to set alarms for specific times associated with a diet plan
- Users can send and receive diet plans to other individuals, medical professionals, and personal trainers.
- Users will have the ability to post and share their diet plans and achievements on social media as well as compete on social media leader boards tied to the app.

Business Benefits

The following lists the business benefits of the application.

- Advertisements within the free version of the app develop ongoing relationships and deals between the business and other companies.
- Application exposure on the market should set a good reputation for the business.
- Competitiveness and ongoing achievements will encourage users to continue using the app.
- The app has potential to stand out of the crowd and become a standard for user-to-professional interaction through creating and sending diet plans: i.e. Doctors and dietitians can give their patients a set diet plan if necessary.
- App has the capability to become a social platform of its own community.

Cost Benefit Analysis

Timeframe

The expected development period for the application is four months. Therefore, all estimates and calculations of the costs of individual components and the total project cost were developed based on this expectation. At the very bottom of this section is a table outlining the project costs and budget.

Team Member Salaries

Firstly, some research was done on the typical income for software developers per year in Australia and then that was broken down further by role. Based on these averages, fair salaries for work required for each role and for the period that each team member will be contracted to work on the application. However, both the UI designer and the marketer will only be contracted to work in the first month and last month of development, respectively. As such, their salaries were based on a one-month period of work, not four.

As the application must provide a support service and must also be maintained with bug fixes and quality of life improvements, two full time team members will be hired to fulfil these criteria as well as provide feature updates to the application.

Equipment Costs

The two members working in UI design and marketing will only be working for one month and not for the full development time and are able to bring their own laptops. They already have the necessary tools to adequately complete their job for the duration of their contract. Three workstations, one for each of the programmers as well as the project manager will be bought alongside all the necessary software tools and licenses.
The budget for the three workstations, including all components and licenses, was decided upon based on a quote given by a local retail store that specializes in PC parts and software.

Ongoing Costs/Reserve

Some of the required functions of the application to be developed need an internet connection to function properly. Because of this, a third party will be used to provide server hosting capabilities and that the cost of that service would be added to the budget (although, it will be charged yearly).

Possible Project Income

Revenue projections were developed based on data collected from several sources; surveys conducted on potential users as well as third party analytical databases. Also observed was the relative success of mobile applications that serve a similar purpose and/or have similar features. It is expected that the application will be downloaded 60,000 times in the first month on the Android platform and 40,000 times on iOS. A further 5,000 full licenses bought for Android and 7,500 purchased on iOS.

If each user logs into the application at least once a day, an expected advertising revenue is $0.1875 on Android and $0.1 on iOS. This is averaged as per user, per day for each platform.
Without accounting for licenses bought for the full version of the application, in the first 18 months of its life time it is expected to make $492,750 in advertisement revenue on Android and $175,200 on iOS. This comes to a total of $667,950 across all platforms.

Project Estimated Budget

Summary of Estimated Development Costs for RPG Health

Wages (over 4-month period)		
	Project Manager	$23,000
	Programmer	$19,000
	Programmer/Documentation	$20,000
	Marketer	$4,750
	UI Design	$4,750
Wages (Full Time)		
	Maintenance/QOL Updates/Feature Updates/Support	$80,000
	Maintenance/QOL Updates/Feature Updates/Support	$80,000
	Total	$231,500
Equipment for Development Team		
	3 x Intel Core i5-7400 1151 Processor	$720
	3 x ASRock B250M-PRO4 LGA 1151	$348
	3 x G.Skill 8G Single PC4-19200/DDR4 2400MHZ 1.20V	$267
	3 x Antec NSK-3100-U3 Micro ATX/Mini-ITX	$177
	3 x Aerocool VX-450 ATX PSU, ATX12V 2.3	$165
	3 x Seagate SkyHawk Surveillance 1TB 3.5in 64MB SATA	$225
	3 x Microsoft Windows 10 Home 64bit USB Flash Drive	$417
	3 x Acer K222HQL 21.5" Monitors	$375
	3 x Logitech MK235 (Mouse + Keyboard Combo)	$96
	Miscellaneous Software Licenses	$1,500
	Total	$4,290
Ongoing Costs/Reserve (Per Year)		
	Server Hosting/Connectivity (charged yearly)	$1,500
	Total	$1,500
Total Budget		**$237,290**

Risk and Feasibility Analysis

Marketing

Creating objectively excellent software that won't be used is a risk when developing health apps. Such apps do not place in the top 50 most used apps in Australia as the list includes mostly tool, entertainment, and social based apps.
To ensure Health RPG is profitable, the app will be presented to users with early builds that incorporate the successful attributes listed above.

Management

Health RPG's development team consists of a UI-designer and two previously independent programmers who are used to utilizing open-source programs to perform their tasks. For this project however, a shared office space and 3rd party software will be used for the duration of the project, this may pose the following management risks:

- The two programmers may be reluctant to work together as independent programmers. This can result in a lack of productivity within the workspace as well as turnover
- The implementation of the new software and seemingly long timeline can cause the developers to take their time during the early stages of developing. This can result in missing the deadline.

To mitigate these risks:

- Extreme programming (XP) will be implemented to allow the programmers to work their own way while collaborating and practicing agile development.
- A constantly monitored iterative development system will be set in place to ensure productivity in the workplace never dips below a satisfactory level.

Resource

A lack of funding and time make for the biggest risks the project will face. The quality and speed of which the developers will function at are highly dependent on these resources and thus pose a plethora of risks:

- While the scope of the project is known to the developers, the client may bring significant changes to the design once development has begun. In the case of an unexpected expansion of the project scope, a budget overrun is likely to occur.
- Incorrect time-estimation on the project manager's part due to the complexity of a critical activity can have a negative impact on the quality of the finished product
- Misunderstanding of the project requirements by any of the team members can halt development of a critical part.

To mitigate these risks:

- Waterfall ideologies will be avoided and planning will be kept to short cycles of development for each function of the project. This iterative development method is better equipped to handle changes in the software requirements and will minimize the risk of falling behind schedule.

- To compensate for budget overruns and ensure the availability of the software, the following non-functional requirements may be reduced in quality:
    - Performance
    - Documentation
    - Stability
    - Maintainability
    - Robustness
- For each team member, any superficial knowledge will need to be discovered and assessed at an early stage during the development to prevent later surprises. A backup will be made available in this case.

Technological

The technological risks involved in the development of the project include:

- The team members lacking the knowledge to operate on the new 3rd party software.
- The customer support member not understanding how the software functions due to a lack of / poor documentation.

To mitigate these risks:

- All the software developers handling the 3rd party programs will be given concise help manuals.
- The second programmer's primary objective will be the documentation and indentation, this will ensure efficient maintenance is possible.

Unfortunately, the possibility of denied access to a 3rd party macro tracking database is a functionality breaking risk. It is impossible to create a database that keeps track of every food product's nutrient information in the given timeframe and if access to an existing database is denied, the software will not be able to track nutrient plans.

Section 2 – Presenting the Requirements

Identifying the Stakeholders

Marketers

The marketer's primary concerns are the functional / non-functional requirements and the effectiveness of the team in terms of time. Performance for example can have a huge impact on the marketer's job of advertising the app.

Customers

The customers' primary concern is profit, this means they care little about non-functional requirements or robustness as their primary concerns are the expenditures and overall effectiveness of the team behind the development. The customer can also have concerns regarding feasibility and risk assessment, progress tracking and schedule estimations.

Maintenance

The maintainer of the app can have many concerns regarding the code and the software it is used in. For starters, it is vital to the individual's job that guidance is available in the form of documentation. Code compatibility is also a concern to the maintainer as language updates get rolled out frequently, this can also be related to non-functional requirement concerns regarding performance and reliability.

End Users

The end user's primary concerns are the performance, reliability, security, behavior, and user-friendliness of the app they will be using. They care about the speed of which the app loads, if any bugs are present within the app, the safety of their personal data inputted into the app and the ability to use a feature effortlessly. Most of their concerns are non-functional requirements.

Architect

The architect's primary concerns are related to requirements traceability. Through the inputted requirements, the architect may also raise concerns regarding the consistent use of the system created. In addition, the architect may show concerns regarding the feasibility of the app once all the requirements have been understood.

Designers

The designer's only concern is requirement traceability as sufficient detail for design is vital for the job and some functional requirements can only be met with an interface.

Developers

The developers' concerns are primarily related to requirement traceability. They are most interested in knowing what the client exactly wants and can express concerns about the functional / non-functional requirements, clear specifications, and the resources available such as hardware and software.

The Requirements

Functionality Requirements

- The ability to create and setup an account with basic starting health analytics
    - i.e. gender, age, weight, height
- To implement a navigation interface. i.e. Home, Build Diet Plan, Message, Manage Account, etc
- The user can plan and preview their own diet plan
- The user will have access to a 3rd party database through Health RPG's search engine API that provides a library of information on most of all available foods
- User ability to track and visualise progress through levels, virtual achievement badges and statistics/graphs
- User ability to send and receive plans to other users
- User ability to change settings
    - i.e. colour, font size
- Users will have access to link their account with major social media websites, can export diet plans to social media and join a leader board system on each website
- Users will have the ability to zoom in and out of their diet plans
- The ability to set alerts aligned with a user's diet plan as friendly reminders
- Allocated memory for saved diet plan(s), diet plan data, message bank and progress should be stored on the user's mobile device.
- Users have access to more features in the paid version than the free version.
    - i.e. Free version will only allow users to store one diet plan whereas the paid version would have the ability to store up to multiple diet plans

Usability Requirements

- The system can support many users connecting to and using the application including the usage of the 3rd party database and messaging system
- Template diet plans are available for users to utilise
- Progress analytics and graphs are presented in order of importance

Reliability Requirements

- System errors should be dealt with and have minimal effects on user experience either with an update and prior to that it should have a temporary solution
- Login authentication should be available 99.9% of the time
- If the 3rd party server malfunctions or experiences downtime, users who are currently making diet plans will have their session stored on their device

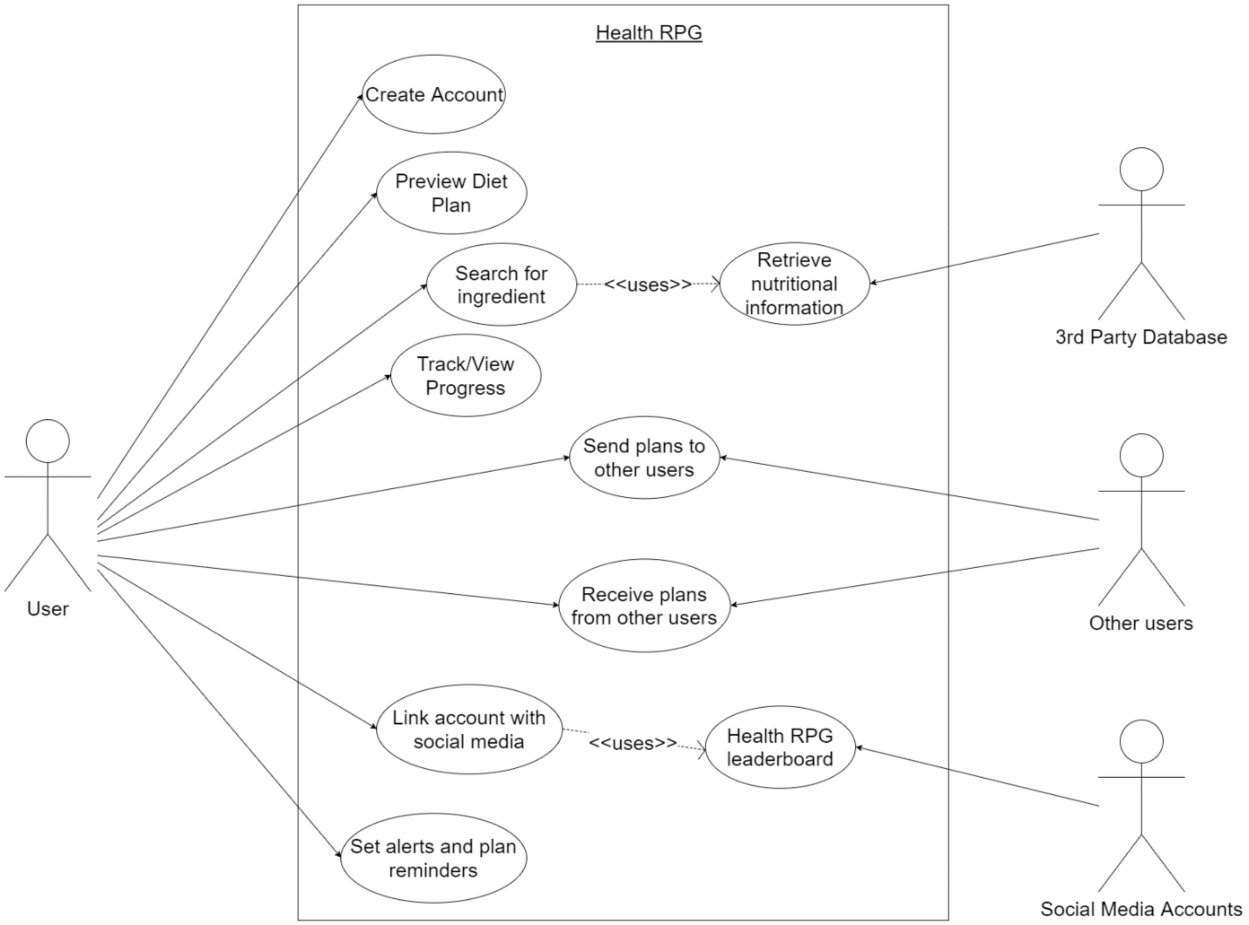
Performance Requirements

- The application should not be very demanding and run as fast and smoothly as possible
- Access to the 3rd party database and social media websites should be quick
- Many users accessing online features should have minimal effect on access time

Security Requirements

- Application login and usage should be safe and secure
- All inputs should be validated and checked by the system upon the user entering information
- Personal user information will be stored on their device reducing the risk of data loss

Requirements in a Use Case Diagram



Use Case Scenario (Sharing plans with other users [Normal Path])

1. The application is in a ready state.
2. The user taps “Preview Diet Plan” button.
3. The application populates screen with the user’s diet plan.
4. User taps “Share [their diet plan]” button.
5. A menu pops up to display sharing options; other users, social media, email, messages, etc.
6. The user selects the option to share with other users.
7. A dialogue box with an empty text field pops up onto the screen.
8. The application prompts the user to enter the mobile phone number of the other user they wish to share their diet plan with.
9. The user enters in the phone number of the other user that they wish to share with.
10. The user hits the “share” button on the dialogue box.

11. A notification is received by the other user via their own Health RPG application to tell them they have received the diet plan of another user.

12. The application populates the screen with the user's diet plan.

Section 3 – Requirement Modelling

Subset of System Requirements

Requirement 1

The user can create and account with their health analytics.

- Constraints:
    - The user must have a unique ID. This could be their Medicare ID number.
    - User must provide their body weight, height, age.

Requirement 2

The user can create their own diet plan

- Constraints:
    - The diet plan must have its components listed in the vitamins / foods database, i.e. the user cannot input an illegal food.

Requirement 3

The user can press a button to preview their own diet plan (**EVENT)**

- Constraints:
    - A diet plan must exist.

Requirement 4

The user can swipe screen and visually examine their dietary progress

- Constraints:
    - A diet plan must exist.
    - Colour blindness support.

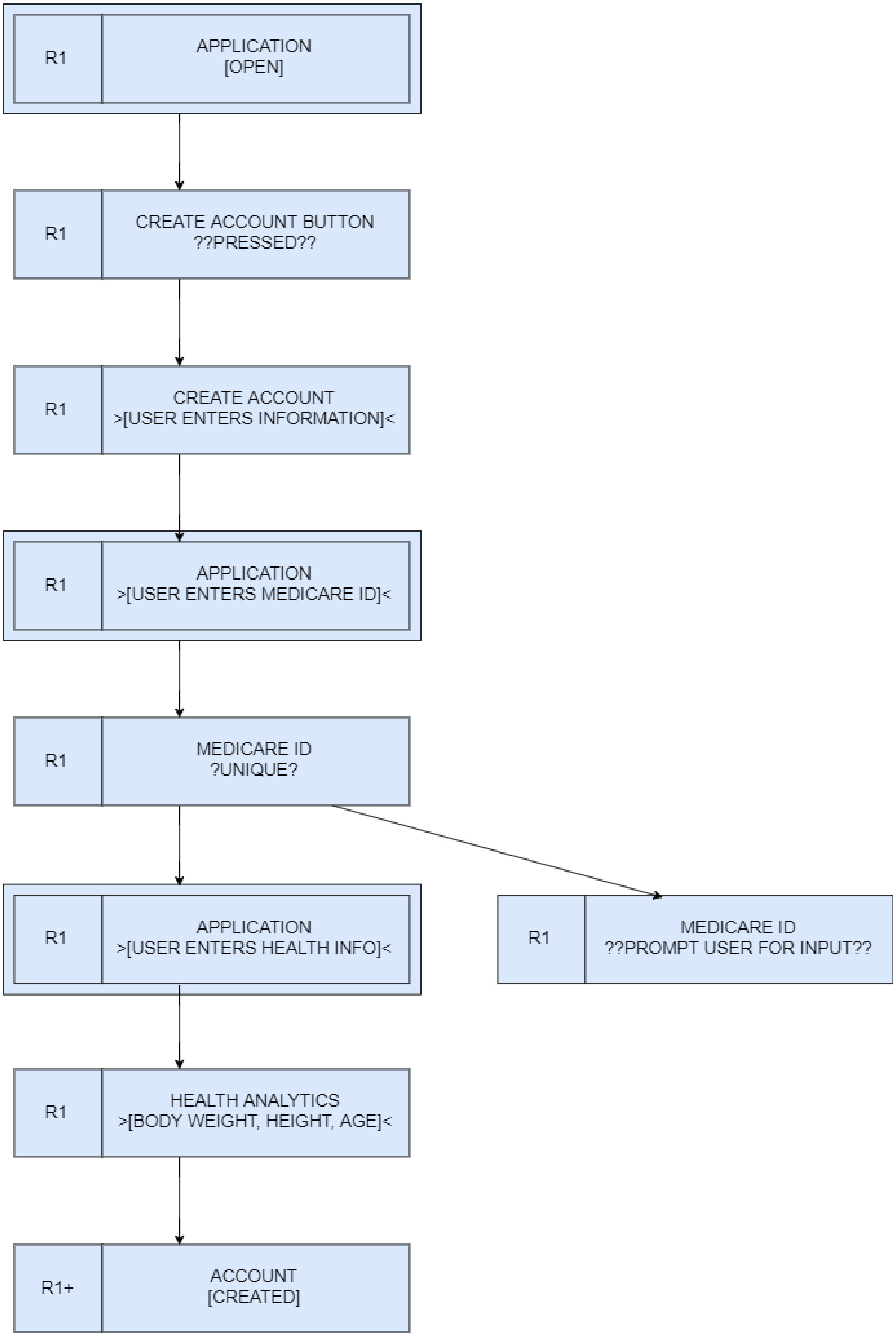
Requirement 5

The user can create a dietary task and set reminders

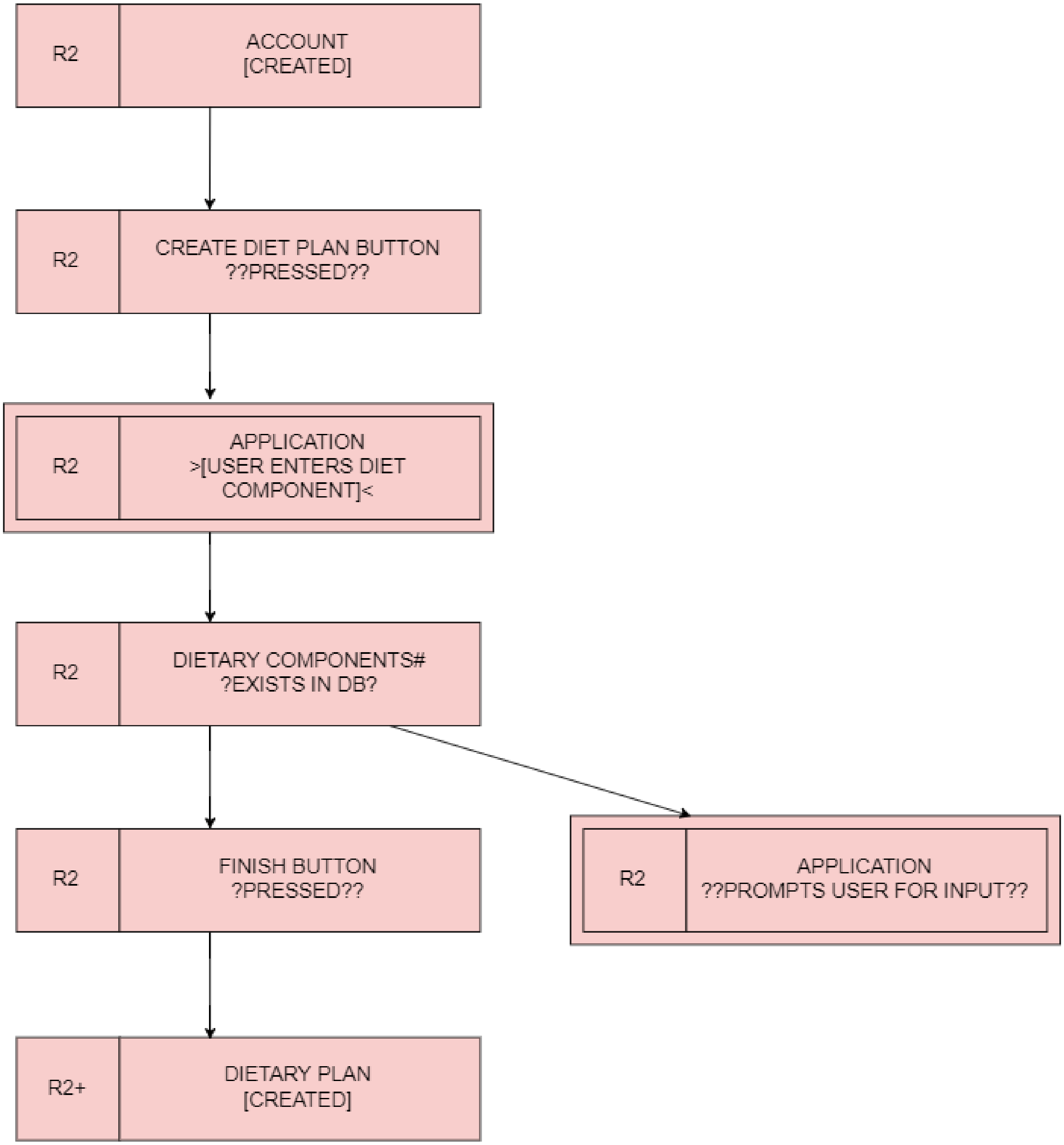
- Constraints:
    - A diet plan must exist.
    - The user has agreed to allow the app to make reminders.

Requirements Models

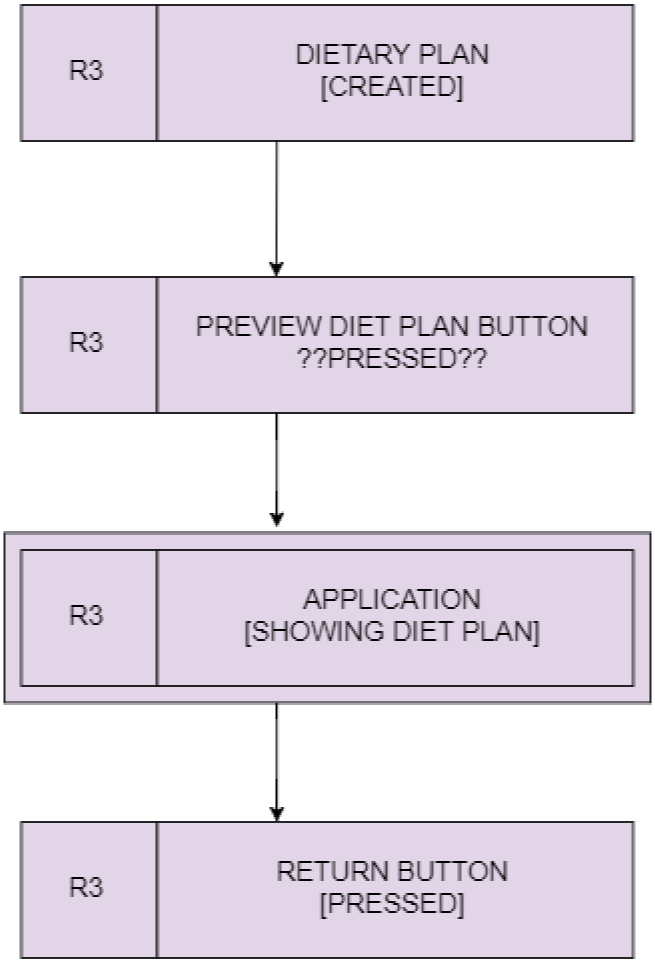
Requirement 1



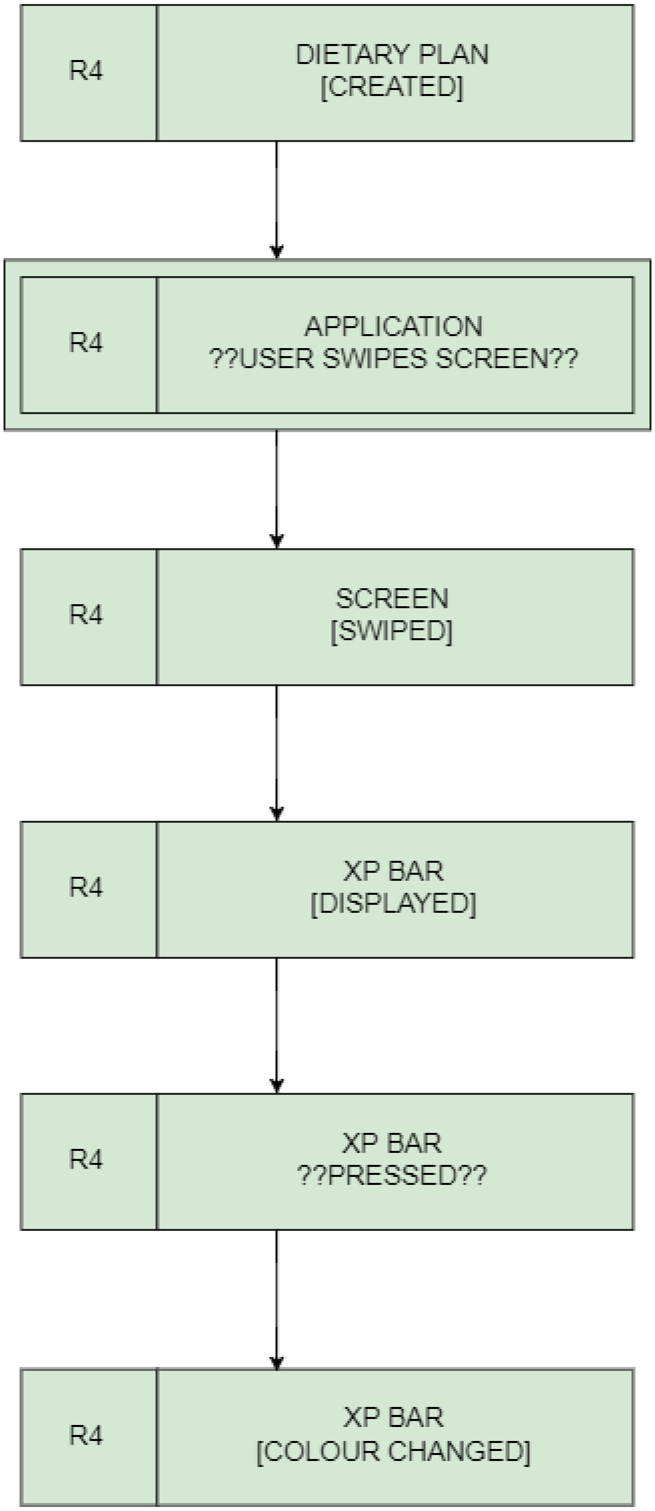
Requirement 2



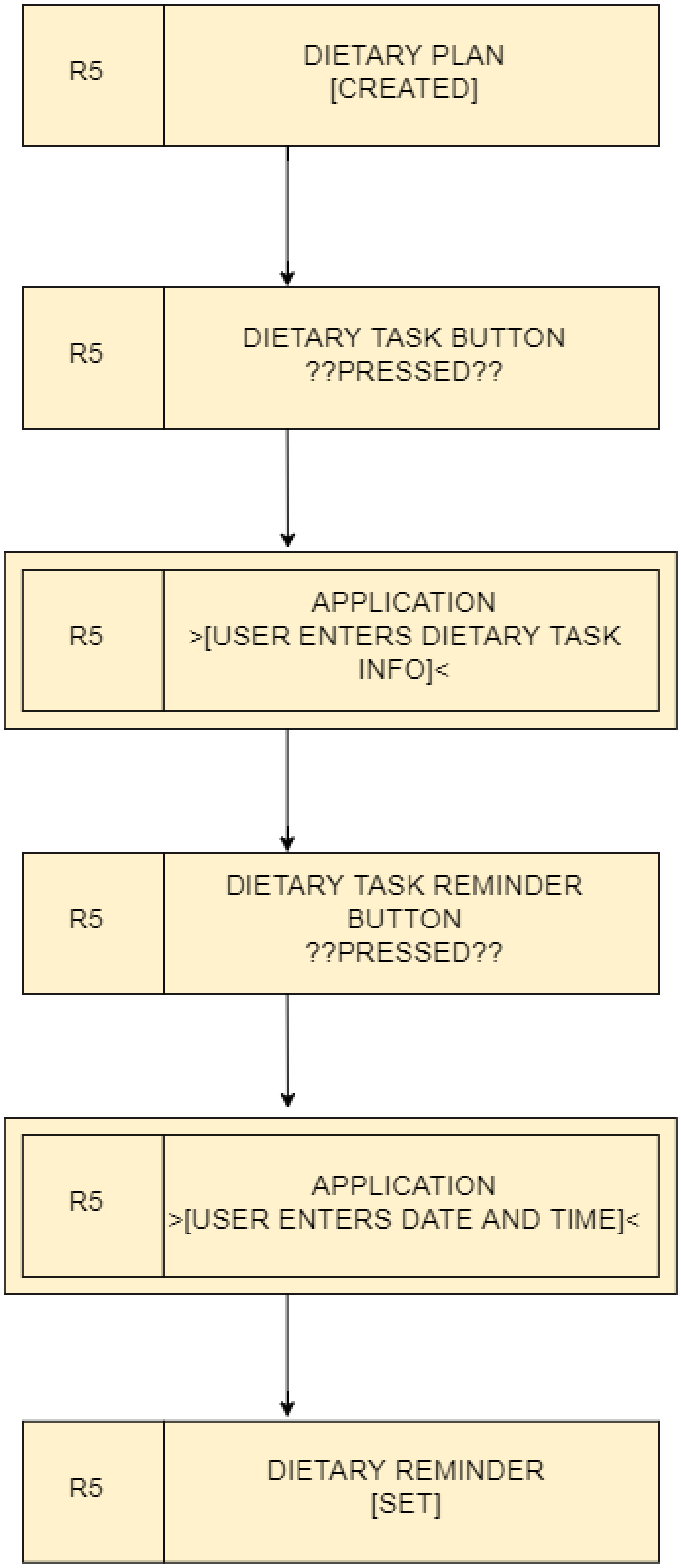
Requirement 3



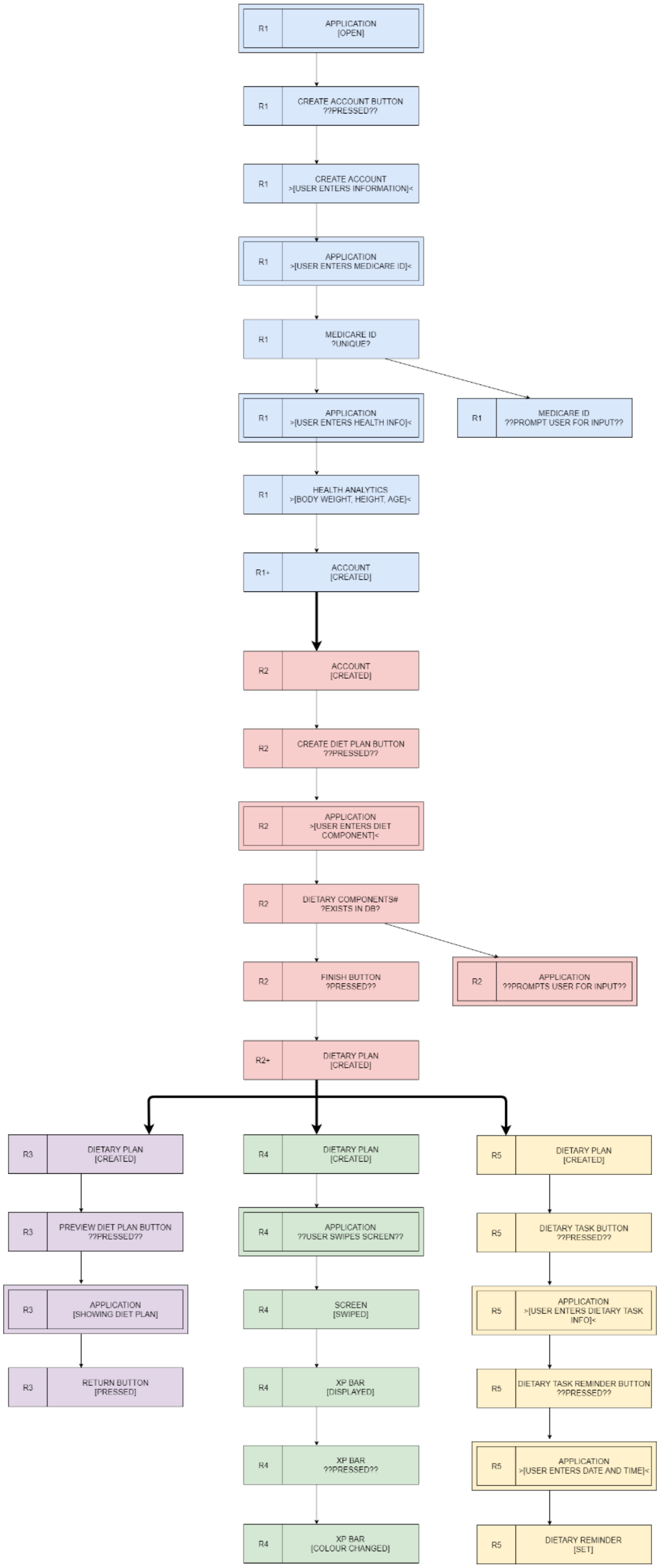
Requirement 4



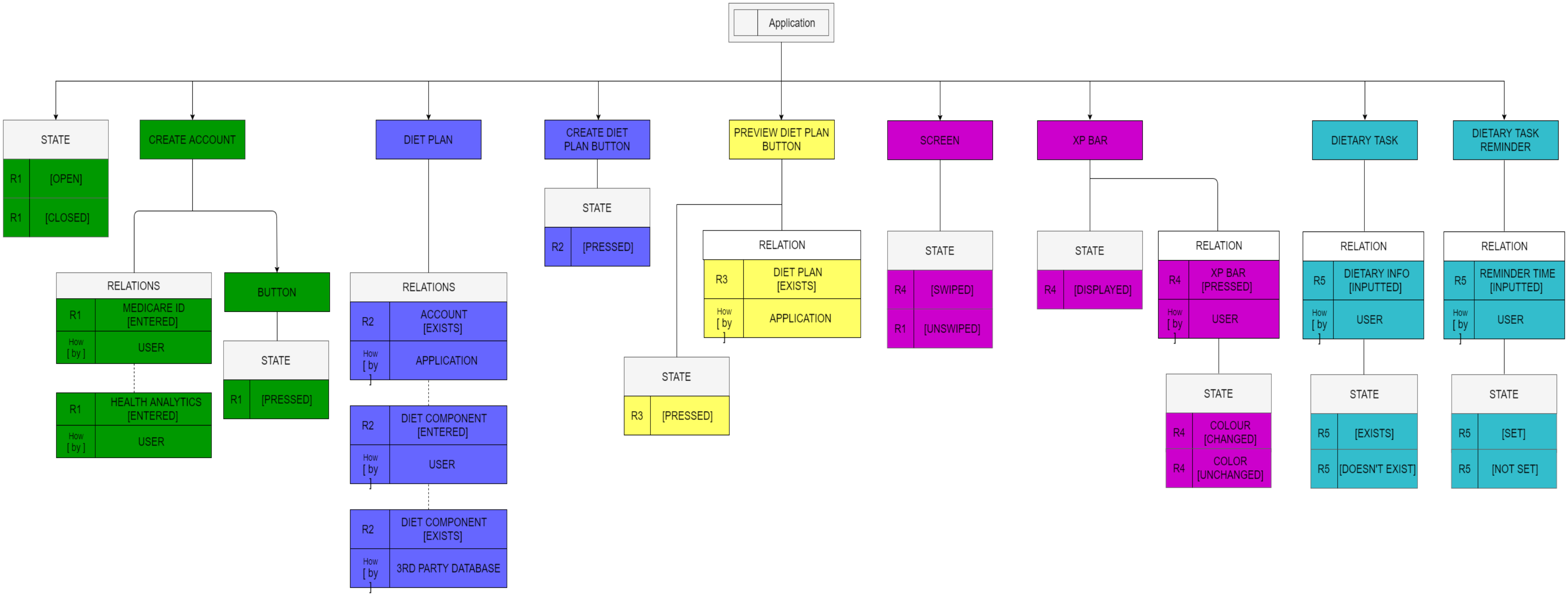
Requirement 5



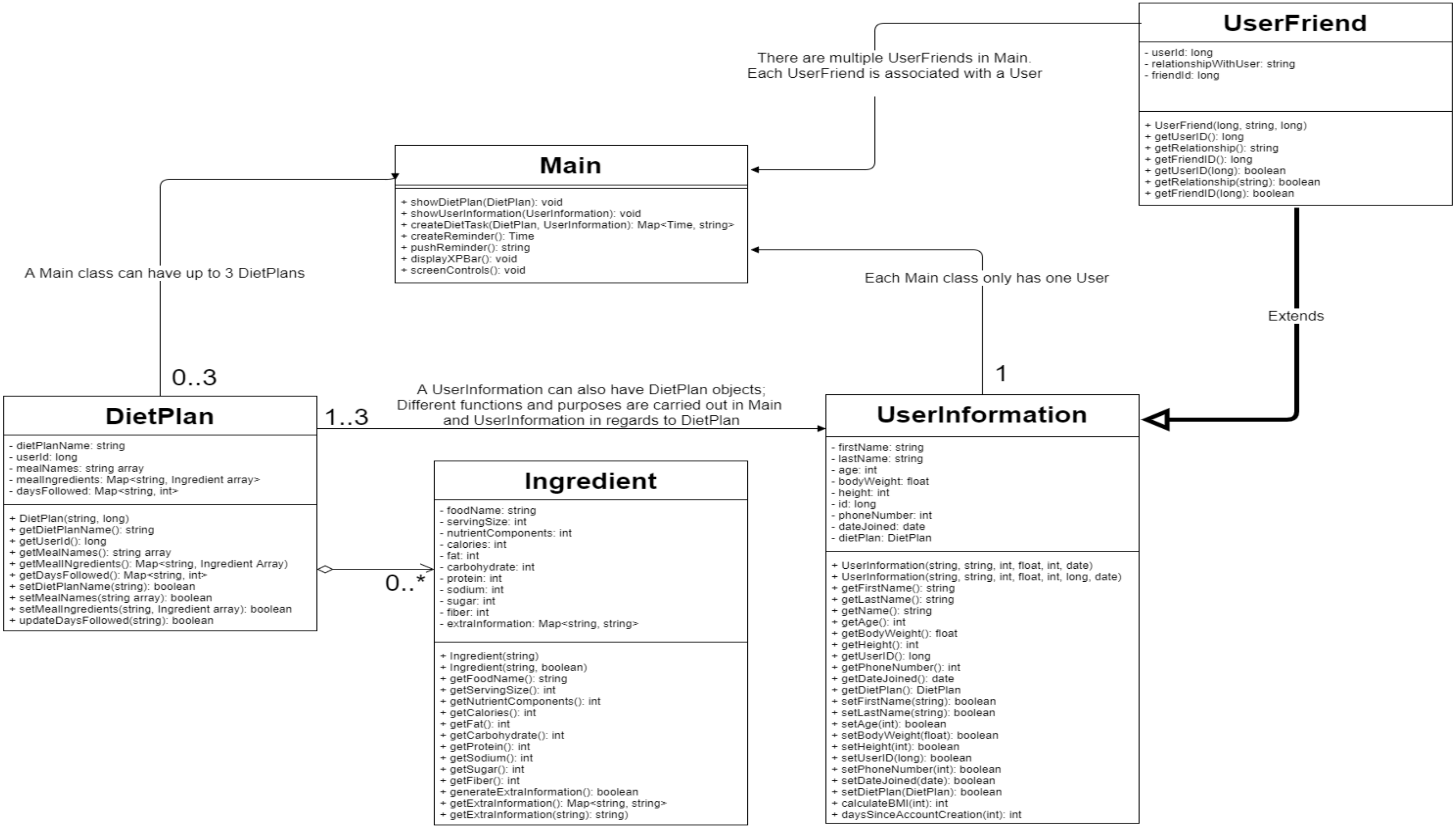
Integrated Behaviour Tree



Integrated Component Tree



Domain Class Diagram


There are multiple UserFriends in Main.
Each UserFriend is associated with a User
A Main class can have up to 3 DietPlans
Each Main class only has one User
Extends
0..3
1..3
A UserInformation can also have DietPlan objects;
Different functions and purposes are carried out in Main
and UserInformation in regards to DietPlan
1
0..*

UserFriend

```
- userId: long
- relationshipWithUser: string
- friendId: long

+ UserFriend(long, string, long)
+ getUserID(): long
+ getRelationship(): string
+ getFriendID(): long
+ getUserID(long): boolean
+ getRelationship(string): boolean
+ getFriendID(long): boolean
```

Main

```
+ showDietPlan(DietPlan): void
+ showUserInformation(UserInformation): void
+ createDietTask(DietPlan, UserInformation): Map<Time, string>
+ createReminder(): Time
+ pushReminder(): string
+ displayXPBar(): void
+ screenControls(): void
```

DietPlan

```
- dietPlanName: string
- userId: long
- mealNames: string array
- mealIngredients: Map<string, Ingredient array>
- daysFollowed: Map<string, int>

+ DietPlan(string, long)
+ getDietPlanName(): string
+ getUserId(): long
+ getMealNames(): string array
+ getMealINgredients(): Map<string, Ingredient Array)
+ getDaysFollowed(): Map<string, int>
+ setDietPlanName(string): boolean
+ setMealNames(string array): boolean
+ setMealIngredients(string, Ingredient array): boolean
+ updateDaysFollowed(string): boolean
```

Ingredient

```
- foodName: string
- servingSize: int
- nutrientComponents: int
- calories: int
- fat: int
- carbohydrate: int
- protein: int
- sodium: int
- sugar: int
- fiber: int
- extraInformation: Map<string, string>

+ Ingredient(string)
+ Ingredient(string, boolean)
+ getFoodName(): string
+ getServingSize(): int
+ getNutrientComponents(): int
+ getCalories(): int
+ getFat(): int
+ getCarbohydrate(): int
+ getProtein(): int
+ getSodium(): int
+ getSugar(): int
+ getFiber(): int
+ generateExtraInformation(): boolean
+ getExtraInformation(): Map<string, string>
+ getExtraInformation(string): string)
```

UserInformation

```
- firstName: string
- lastName: string
- age: int
- bodyWeight: float
- height: int
- id: long
- phoneNumber: int
- dateJoined: date
- dietPlan: DietPlan

+ UserInformation(string, string, int, float, int, date)
+ UserInformation(string, string, int, float, int, long, date)
+ getFirstName(): string
+ getLastName(): string
+ getName(): string
+ getAge(): int
+ getBodyWeight(): float
+ getHeight(): int
+ getUserID(): long
+ getPhoneNumber(): int
+ getDateJoined(): date
+ getDietPlan(): DietPlan
+ setFirstName(string): boolean
+ setLastName(string): boolean
+ setAge(int): boolean
+ setBodyWeight(float): boolean
+ setHeight(int): boolean
+ setUserID(long): boolean
+ setPhoneNumber(int): boolean
+ setDateJoined(date): boolean
+ setDietPlan(DietPlan): boolean
+ calculateBMI(int): int
+ daysSinceAccountCreation(int): int
```