**Requirements**

**Requirements Gathering Summary:**

In order to make our system as useful as possible, we decided to collect our requirements from a wide range of varying resources. Primarily, we researched heavily into the area surrounding Personal Informatics and based a large portion of our requirements off this. By gathering a lot of requirements from articles such as those that have been peer reviewed allows us to have a better understanding about the sorts of things that our final system should and shouldn’t be doing.

Moreover, we held a focus group. During this focus group, several requirements that we had missed were made apparent. By having a semi-structured conversation with potential stake holders, we were able to see roughly what a large proportion of people would want from a system like the one that we are developing. In addition to this, the focus group allowed us to quickly and efficiently gather data from people who most greatly reflect the audience we are aiming for. This meant that we had more time to work on constructing the system as opposed to researching for more requirements.

Finally, we also sent out a questionnaire. We received X RESPONSES, from which we were able to gather additional requirements. From the focus group, a mass of information was yielded which showed us a lot of perspectives we hadn’t necessarily put a lot of thought into. It also provided many niche requirements which if we have time to get to would really enrich our system and take it a step above where we were aiming before.

**Requirement Priorities:**

Upon reviewing all the data we had gathered to compile our list of requirements, we labelled all of our requirements with priorities and dependencies. The dependencies really highlighted how crucial some requirements were and which of our requirements were more of the additional extras which would be nice to include if time allows. We considered the necessity of each requirement and its dependencies for the priority of any given requirement.

**Requirement Conflicts:**

When we had created our list of requirements, we were careful to avoid any conflicting requirements. These could have arisen from varying opinions from different stake holders. In order to avoid including conflicting requirements, we considered all the data that we had collected and then prioritised the one with the highest interest. Finally, we split all of our requirements into a structured indexed list to help illustrate dependencies and also to avoid repetition or conflict between requirements.

**Functional Requirements**

|  |  |  |
| --- | --- | --- |
| ***1. Requirement Name - Server*** | | |
| 1.1 | Send data to client when requested | Author: Sam Davidson |
| **Description:** The web app must attempt to send data to the client when requested. This data could be specified so it needs to be able to handle the specific request. The web app will then deal with the data it is given. | | Priority: HIGH  Dependencies: 2.1 |
| ***2. Requirement Name - Data Collection*** | | |
| 2.1 | Get data from Spotify | Author: Joseph Cryer |
| **Description:** The server must be able to request data from Spotify’s public web API for a user. This can only happen once this user has connected to Spotify through our system successfully.  Our system must send a message to the Spotify web API containing the authorisation token for that user (see 'Connect to Spotify'). On success, Spotify's web API will return an access token that can then be used to make requests to the API for a certain amount of time. | | Priority: HIGH  Dependencies: None |
| ***3. Requirement Name - Data Storage and Processing*** | | |
| 3.1 | Store data from Spotify for each user | Author: Teodora Dinca |
| **Description:** The server must process the data received from the Spotify API and take all the data that will be used and put it into a database in an efficient intermediate format keeping only the data we need and not storing any redundant data which offers little to no value to our client. | | Priority: HIGH  Dependencies: 2.1 |
| 3.2 | Recommend songs based on trends | Author: Fraser Dwyer |
| **Description:** The system should be able to recommend songs to the user based off of mood patterns or based off of genre that the user likes to listen to. | | Priority: MEDIUM  Dependencies: None |

|  |  |  |
| --- | --- | --- |
| ***4. Requirement Name - Client*** | | |
| 4.1 | Create a new account | Author: Sandra-Maria Corradi |
| **Description:** The server must allow the user to create a new account. The user will be required to enter a username and a password when creating an account. The information will be sent to the server which will check it against the existing users’ usernames. If the username they entered already exists, the user will receive an error asking them to enter another username. If the username they entered is valid, a new account will be created by adding the new user’s information to the users list (the username and the hashed password). A new file will be created and linked to the user in which future data will be stored. The user will receive a message saying that their account has been created. | | Priority: HIGH  Dependencies: 3.1 |
| 4.2 | Connect to Spotify | Author: Joseph Cryer |
| **Description:** The client must request that the user connects their account to their Spotify account. The user will be redirected to a Spotify webpage showing the data that is being requested by our system, and will be asked to first login to Spotify, and then authorise our system to use the relevant data. If successful, a 'success' callback message will be sent to the server containing an authorisation code, meaning our system can now make requests to the Spotify web API to request that user's music data. If unsuccessful, a 'fail' callback message will be sent to the server. | | Priority: HIGH  Dependencies: |
| 4.3 | Login to the server | Author: Sandra-Maria Corradi |
| **Description:** The server must allow users to log into their accounts. The user will need to enter their username and password. The server will then try to find the given username in the usernames list. If it is not found the user will receive an error. If it is found, then the supplied password will be hashed and compared with the stored hash. If the password is correct the user will be logged in and will have access to his data, otherwise the user will receive an error. | | Priority: HIGH  Dependencies: 1.1, 3.1 ,4.1 |
| 4.4 | Display data to the user | Author: Teodora Dinca |
| **Description:** Reformat data into a set of interesting visual representations which can be displayed to the user, data for these visual representations should be provided only when they are actually requested by the user so as to save the user bandwidth and reduce server load, these should also follow the guidelines for visual data as set out in the research and each should have a specific aim such that they provide the user with understandable and usable data | | Priority: HIGH  Dependencies: 1.1, 2.1, 3.1, 4.2, 4.3 |
| 4.5 | Request data from server | Author: Sam Davidson |
| **Description:** The web app must be able to request data from the server for whatever process it would like the data for. This data should be able to be requested for specific data, for example, song details, song name or even everything in a database, etc. This is useful as it can restrict the amount of data to be sent over to the network. | | Priority: HIGH  Dependencies: 2.1, 4.3 |

**Non-Functional Requirements**

|  |  |  |
| --- | --- | --- |
| **1 – Requirement Name: Scrum Methodology** | | |
| **1.1** | We must use the Scrum method in order to develop our project | Author: Fraser Dwyer |
| **Description:** Must use the agile development technique called ‘Scrum’. This involves regular ‘Sprint Meetings’, following sprint backlogs, and sprint reviews in order to gauge sprint progress. This development technique works well for obtaining rapid feedback. This is vital in a project like the one we are producing since our goals are forever changing with elements being implemented and tested and then potentially removed or altered. | | Priority: HIGH  Dependencies: 1.2, 1.3, 1.4 |
| **1 - Requirement Name: Number Of Sprints** | | |
| **1.2** | We must have at least 3 sprints | Author: Fraser Dwyer |
| **Description:** In order for our project to be successful, having many sprint meetings will be vital. Sprints allow us to have target focussed goals allowing for increased productivity. These sprints will also present developments in terms of what needs to be done in the next sprint by throwing up new issues which can be added to the product backlog and dealt with in the next sprint. | | Priority: HIGH  Dependencies: 1.3 |
| 1. **- Requirement Name: Sprint Duration** | | |
| **1.3** | Sprint duration must be between 1 and 3 weeks | Author: Fraser Dwyer |
| **Description:** The time we have been allocated for our project allows us between 1 and 3 weeks for a given sprint in order to fit in 3 sprints before the deadline. We must have at least 3 sprints as this allows us to review and make changes to any given approach we may be taking. | | Priority: HIGH  Dependencies: 1.2 |
| **1 – Requirement Name: Review Functional Requirements** | | |
| **1.4** | We must regularly review our functional requirements | Author: Fraser Dwyer |
| **Description:** We must regularly review our list of functional requirements after each sprint. This is important since each sprint will reveal changes that need to be made to our list for one reason or another such as software/hardware limitations, time restrictions, etc. | | Priority: HIGH  Dependencies: None |
| **1 – Requirement Name: Risk Managements** | | |
| **1.5** | We will incorporate risk management into our software process | Author: Fraser Dwyer |
| **Description:** We will be using risk management assessments in order to ensure our project is following a maintainable development design. For example, we will avoid at any cost to develop a server in a language only one person knows in case that person happens to fall ill. This would be a very big setback to the project and would halt progression massively. | | Priority: MEDIUM  Dependencies: None |
| **2 - Requirement Name: Altering Requirements** | | |
| **2.1** | After researching, we will add to our requirements | Author: Fraser Dwyer |
| **Description:** Throughout the process, we will be researching our chosen area of personal informatics. Due to this, we will gain more knowledge around our specific area of PI. We will use this continually growing understanding around our area of PI and use it to adapt and add to our list of initial requirements. | | Priority: MEDIUM  Dependencies: None |
| **3 – Requirement Name: >= 4 Core PI Articles** | | |
| **3.1** | We will read and cite at least four core articles in the area of PI | Author: Fraser Dwyer |
| **Description:** We will read and use at least four core articles in the area of PI in our project. This is so our project is well informed and knowledgeable surrounding the area of PI. This will also aid our project in terms of what to aim for what our project should and shouldn’t do. | | Priority: MEDIUM  Dependencies: None |
| **3 - Requirement Name: 8 Articles of Any Kind** | | |
| **3.2** | We will include 8 articles of any kind related to PI in our project | Author: Fraser Dwyer |
| **Description:** We will include these articles surrounding Personal Informatics as research within our project in order to have a wider range of knowledge and understanding regarding the project domain. | | Priority: MEDIUM  Dependencies: None |
| **3 – Requirement Name: Peer Reviewed Core Articles** | | |
| **3.3** | Our >= 4 core articles must be peer reviewed articles | Author: Fraser Dwyer |
| **Description:** By including peer reviewed articles within our project ensures that the articles that we are citing have been validated and deemed to be accepted as factual. This is important as it means that we are using scientifically accepted tests and data as part of our own project. | | Priority: MEDIUM  Dependencies: 3.1 |
| **3 – Requirement Name: Additional Research** | | |
| **3.4** | We will gather other research materials of varying qualities | Author: Fraser Dwyer |
| **Description:** By gathering other articles, not necessarily peer reviewed, means that we vastly broaden our range in terms of people’s studies and findings. Regardless of these articles’ quality, they will be useful to us by letting us see more than one perspective on certain things related to PI. | | Priority: LOW  Dependencies: None |
| **4 – Requirement Name: Data Usage** | | |
| **4.1** | Must use users’ data for allowed purposes only | Author: Fraser Dwyer |
| **Description:** We must only allow for the user’s data to be used for the intended purpose. Their data will not be distributed to third parties and it won’t be used without the user’s consent. | | Priority: HIGH  Dependencies: None |
| **4 – Requirement Name: Data Storage** | | |
| **4.2** | Data must be kept securely | Author: Fraser Dwyer |
| **Description:** The user’s stored data will be kept secure by encrypting it and securing it on our server. Only authorised persons will be allowed to access it to prevent data being stolen or edited. | | Priority: HIGH  Dependencies: None |
| **4 – Requirement Name: Data Collection** | | |
| **4.3** | The user must have given permission for their data to be collected | Author: Fraser Dwyer |
| **Description:** The user must agree for any of their data to be collected and stored before it is recorded. The user must also be given permission to restrict the data that is collected about them. | | Priority: HIGH  Dependencies: None |

|  |  |  |
| --- | --- | --- |
| **5 – Requirement Name: Usability** | | |
| **5.1** | The software system must be easy to navigate and obvious to use. | Author: Fraser Dwyer |
| **Description:** This is to maximise the use each user gets out of using the software. By making the system easy to use, users will be able to access its full potential. | | Priority: HIGH  Dependencies: None |
| **6 – Requirement Name: Error Testing** | | |
| **6.1** | The system should carry out scheduled error tests | Author: Fraser Dwyer |
| **Description:** This must ensure that the system, both hardware and software components are functioning as they should be and that the users’ stored data has not been compromised. | | Priority: HIGH  Dependencies: None |
| **6 – Requirement Name: Test Plans** | | |
| **6.2** | The system must pass a series of test plans | Author: Fraser Dwyer |
| **Description:** The system produced must undergo a series of test plans since test-driven development is a great way to find unexpected outputs and bugs which otherwise may go undetected. Evidence of this must be provided. | | Priority: HIGH  Dependencies: None |
| **6 – Requirement Name: State and Show Clear Hypotheses** | | |
| **6.3** | Our hypotheses must connect claims found in research to observable behaviours in the finished system. | Author: Fraser Dwyer |
| **Description:** This is so that our final product can be seen to have been based off scientifically correct research and that our project has achieved what we set out for it to do. | | Priority: HIGH  Dependencies: None |
| **6 – Requirement Name: Analyse Empirical Data** | | |
| **6.4** | We must analyse the data generated from users using our finished system. | Author: Fraser Dwyer |
| **Description:** We must analyse data generated by our system using valid statistical methods to discover any trends or patterns within our collected data. | | Priority: HIGH  Dependencies: None |
| **6 – Requirement Name: Error Reporting** | | |
| **6.5** | The system must generate error reports upon crashing | Author: Fraser Dwyer |
| **Description:** The system must generate user friendly error messages to users depending on the reason as to why the system failed. The system should also send detailed error reports back to the developers so that corrections can be made to the system if necessary. | | Priority: HIGH  Dependencies: None |
| **6 – Requirement Name: Interface Reliability** | | |
| **6.6** | The user interface must be tested thoroughly | Author: Fraser Dwyer |
| **Description:** The user interface must be thoroughly tested against things such as invalid user input and error handling to minimise system failures. | | Priority: HIGH  Dependencies: None |
| **7 – Requirement Name: Customisation** | | |
| **7.1** | The system should offer some customisation options | Author: Fraser Dwyer |
| **Description:** The system should be able to offer the user varying levels of customisation for displaying their data. For example, offering light or dark mode. | | Priority: LOW  Dependencies: None |