**COS60010 - Technology Inquiry Project**

**Semester 1 - 2024**

**Deliverable 2**

**Group Project Concept Report**

**Group Name: GROUP 1**

**Student members: Arun Ragavendhar Arunachalam Palaniyappan - 104837257**

**Christopher Opie – 105380202**

**Layan –**

**Amiraj –**

**Henil –**

**Workshop: Thursday – 14:30-16:30 -Room EN207**

**Facilitator: Dr.Alfandi Yahya**

**List of content**

**Topic Pg. No**

**Acknowledgement to Country---------------------------------------------------------------------------- 3**

**1.Introduction------------------------------------------------------------------------------------------------- 4**

**2.Proposed Features----------------------------------------------------------------------------------------- 5**

**3.Key Technical Decisions---------------------------------------------------------------------------------- 7**

**4.Design--------------------------------------------------------------------------------------------------------- 9**

**5.Timeline------------------------------------------------------------------------------------------------------ 18**

**6.Risk Management----------------------------------------------------------------------------------------- 19**

**7.Conclusion--------------------------------------------------------------------------------------------------- 21**

**8.References--------------------------------------------------------------------------------------------------- 22**

**Total Word count** (excluding the report title page, list of content, :

Acknowledgment to country and references)

**Acknowledgement to Country**

Today’s modern-day Melbourne and Swinburne University of Technology is situated in what was originally the Kulin Nation of the past. As Swinburne students, who are truly grateful and proud to study at this esteemed institution, we would like to humbly pay my deepest respects to the Wurundjeri People of this nation who are the traditional owners of these lands.

Additionally, we would also like to sincerely thank the students, alumni, partners, and guests of Swinburne who are from Aboriginal and Torres Strait Islander backgrounds.

It is our honour and we feel proud to recognise and acknowledge the link of spirituality, history and culture of this place to the Wurundjeri country.

**1.Introduction:**

This proposed Chemistry Quiz App seeks to provide a unique method to enhance conventional classroom education by giving students an entertaining platform to improve their understanding of chemical science principles. Digital resources are becoming increasingly more significant for engaging and educating the younger generation. There is a constant demand for easily accessible and interactive learning tools. In light of this, the Chemistry Quiz App aims to provide a more hands-on learning experience than the traditional approach of books and lectures for learning fundamental chemistry principles. The app is designed to facilitate a deeper understanding of tricky chemistry concepts, as students will be faced with innovative questions and are offered instant feedback. It is developed especially for students in their final years of high school, catering both to those who are starting to discover the field of chemistry, as well as more experienced learners through optional difficulty settings.

A game-like experience is offered to students by presenting them with their previous statistics such as their high score, as well as the current leaderboard, to make the learning experience more enjoyable and rewarding. A colourful, user-friendly and accessible user interface would be offered to make the app suitable for a large audience. Staff are provided the ability to add students or modify their stored details via a separate administrative area. They would also be provided the ability to add additional questions and teaching resources of their choosing. Owing to these features, the app can stay relevant and up to date with the latest developments in chemistry, thus we envision that the app would become a and remain a significant learning resource for budding chemistry students.

**2.Proposed Features**

**2.1 Establishing quiz functionality (displaying questions and collecting answer inputs) for chemical structure-based chemistry questions**

Priority: 1 (Highest)

A student should have a series of chemistry-based quiz questions displayed to them, and have their answers accepted by the application. We are focussing initially on chemical structure-related questions because it is fundamental to many branches of chemistry, and is a challenging topic for students. Since it is not a trivial idea to implement, its creation will help distinguish the program from existing chemistry-based quizzes.

**2.2 Displaying a list of correct and incorrect answers at the end of the quiz**

Priority: 2

In order to gain the most value from playing the game, players should be given feedback at the end of a game which displays the questions asked, their responses, and the correct answers if need be. Students will be encouraged to revisit their textbooks or notes, in order to clear up any misunderstandings they may have.

**2.3 Incorporating username/password-based login & authentication for students**

Priority: 2

Students should be able access the Chem Quiz lobby by inputting their username and password which has been provided to them. We envision that, in the future, the app could be accessed though Instatute’s Learning Management System, in which case they would be able to use the same credentials.

**2.4 Recording and retrieving player statistics and scores via the database**

Priority: 2

Students should have their scores saved into the database after they attempt a quiz. Statistics from their record of their previous attempts should be provided to them after they log in to the application, such as their highest score or their previous attempts quiz attempts in graphical format, as well as a leaderboard of the top students. This is designed to show students their improvement over time and provide a sense of accomplishment. At the end of each session their score should be immediately updated with the latest attempt, and they should be provided with updated statistics.

**2.5 Deploying the project to the web**

Priority: 3

The application should be deployed to the web as soon as the higher priority features have been implemented. Since the development environment is different to the production environment, an early deployment will provide ample time to sort out any bugs that arise as a result. At this stage, the project may be considered to be the minimum viable product.

**2.6 Providing admin access to user-friendly database modification APIs for teachers**

Priority: 4

Teachers and staff should be able to have access to an administrative area of the Chem Quiz application. Here they should be provided the ability to perform tasks such as the addition, deletion or updating of quiz questions and student information, or resetting the leaderboard. The addition of questions feature specifically would allow teachers to provide a more customised experience for their students, ideally resulting in questions being directly relevant to the students' current studies.

**2.7 Implement difficulties levels for questions**

Priority: 4

Students should be able to modify the difficulty level of the quiz questions to be asked, since we are aiming to have this application be used by at least high school students from a range of year levels. This may result in having different leaderboards for the different difficulty levels.

**2.8 Extending the quiz with additional question types**

Priority: 5 (Lowest)

Students should be tasked with answering a range of different types of chemistry questions. This adds diversity to the questions and challenges to keep the students engaged, and will allow for a wider subject scope to be covered. Some additional question types include: naming the type of chemical reaction shown, naming the highlighted functionality on a chemical structure, counting the number of a particular type of atom on a provided structure, and finding the mistake in an incorrectly drawn structure.

**2.9 Adding a timer for questions for an added challenge**

Priority: 5

Students may be restricted to having only a limited time to solve each question. This feature seeks to emulate an exam-like experience, which may better prepare the students for their acutal exams. It may also discourage cheating during the quiz to gain better scores, since there would not be enough time to look up the particular questions.

**3. Key Technical Decisions:**

**3.1 User-Interface:**

HTML/CSS/JavaScript

For an interactive study tool, the best method to engage with users is through a graphical or visual medium. The combined front-end web technologies HTML, CSS and JavaScript provide a developer-friendly method of creating user interfaces. Web-based projects also have the benefit of being accessible to a large number of users, as a properly deployed program can accessed by any device with an internet connection.

Vanilla JavaScript

Due to the constraints of the knowledge of the team, we have decided to refrain from using a frontend library/framework such as React or Vue, and will use JavaScript instead of TypeScript. Additionally, a frontend library/framework would provide a lot of features that we have no use for, adding a lot of unnecessary code and complexity to the application. This minimalist approach will lead to smaller file sizes, which means faster initial loading times.

**3.2 Backend: PHP**

A range of backend languages would be suitable for our application, as the primary concern to provide an interface with our database. In keeping with our minimalist approach, PHP is a strong candidate because it can provide this interface, it is lightweight, and it doesn't have complicated syntax. We prefer what it offers over the complicated syntax of Java, and over Python, which would require the use of a backend framework such as Django or Flask. As with the user interface, these frameworks would add unnecessary code and bulk to our applications, whereas PHP can be used as is.

**3.3 Database: MariaDB**

We have decided to store our data in a relational database using SQL, since the entries to the database will either be curated (e.g. quiz questions, specific information about chemicals) or at least structured (scores, users). Additionally, the volume of data is will not be great enough to require a NoSQL approach. The relational database management systems MySQL and MariaDB integrate with PHP easily and members of the team are familiar with its syntax, which make them stand-out candidates. SQLite3 is another option but it comes with the downside that it is single-threaded, thus can only handle a limited number of concurrent read and write operations - which is perhaps not suitable for an online game.

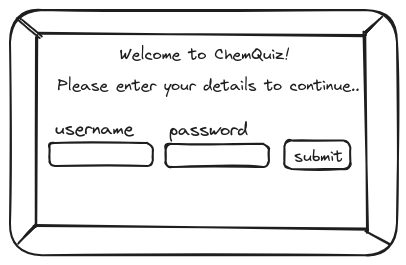
**3.4 Deployment**

Since we are developing a web-based project, the intention is to initially deploy the project either to a virtual private server (VPS), to the Mercury server at Swinburne, or to an online hosting service, before being ideally integrated with the Instatute online platform. In the case that the project needs to run on a local machine, the source code and a database dump file will be provided, in addition to documentation on how to set it up.

**4.Design and in-depth view of the proposed features**

**4.1 User-facing pages and their functionalities**

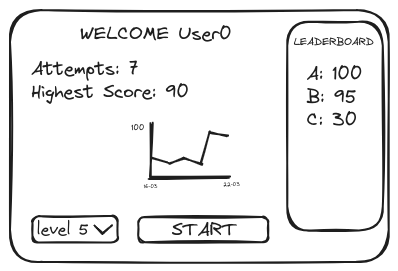
**4.1.1 Login Page:**

**Fig x: User Interface for the login page**

Upon clicking the submit button, and after client-side validation, the inputted username and password are sent to the server. The server firstly sanitises the input, then queries the database using the username, and finally retrieves a hashed version of the user's password. The server hashes the inputted password and compares it with the stored version. If the two don't match, or if the username was not located on the database, a failure response will be sent to the client, and the user will be prevented from proceeding. They will be prompted to check their credentials.

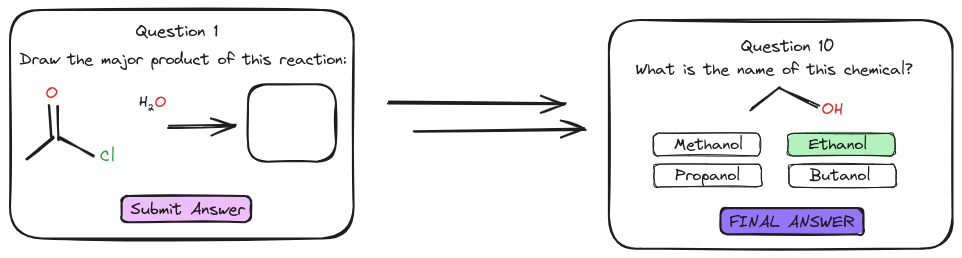
If they do match, then the user has been validated. If the user is an admin, they will be directed to the administrative area. However, if the user is a student, the server will retrieve the complete record of the student's quiz attempts, as well as the top scores of other students, via SQL queries. This data is sent to the client as Json for rendering the welcome page.

**4.1.2 Welcome Page:**

Fig x: User Interface for the welcome lobby

After clicking the start button, the selected difficulty is sent to the server where is incorporated in a SQL query to the database, to retrieve 10 questions of appropriate difficulty. The questions are sent to the client in one batch as JSON for rendering the questions pages, preventing the need for network requests after each response ~~and providing a smoother interface~~.

**4.1.3 Questions Pages:**

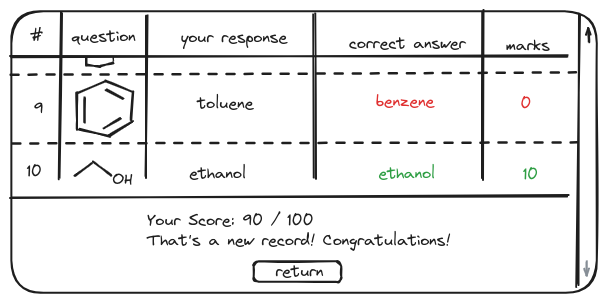
**Fig x: User Interface for the questions pages**

Once a question has been answered, it gets stored as an array, and the next question is dynamically rendered to the screen using JavaScript ops. Once an answer to the final question is submitted, all responses in the array are sent to the server, alongside an identifier for the student. The server sends a sql query to the database, retrieving the correct answers to all questions asked. It determines the student's score, and stores on the database using a SQL update query. The questions, answers and scores are sent to the client as JSON.

**4.1.4 Results Page:**

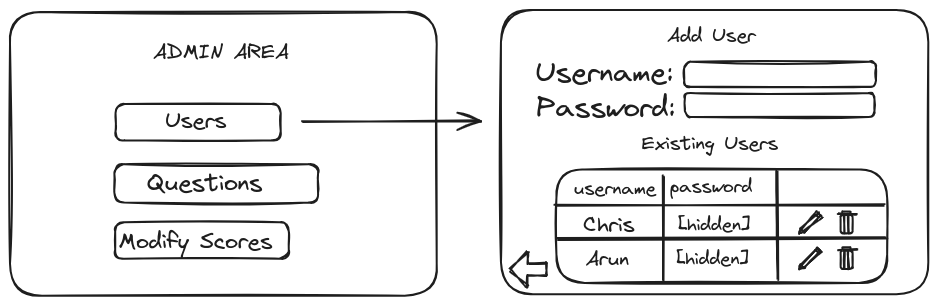
User entry

User entry

**Fig x: User Interface for the results page**

Clicking the button will cause a re-render of the screen to the welcome page, using the fresh and up-to-date scores information that was retrieved earlier.

**4.1.5 Admin Area:**

Fig x: Example of a user Interface for the admin pages

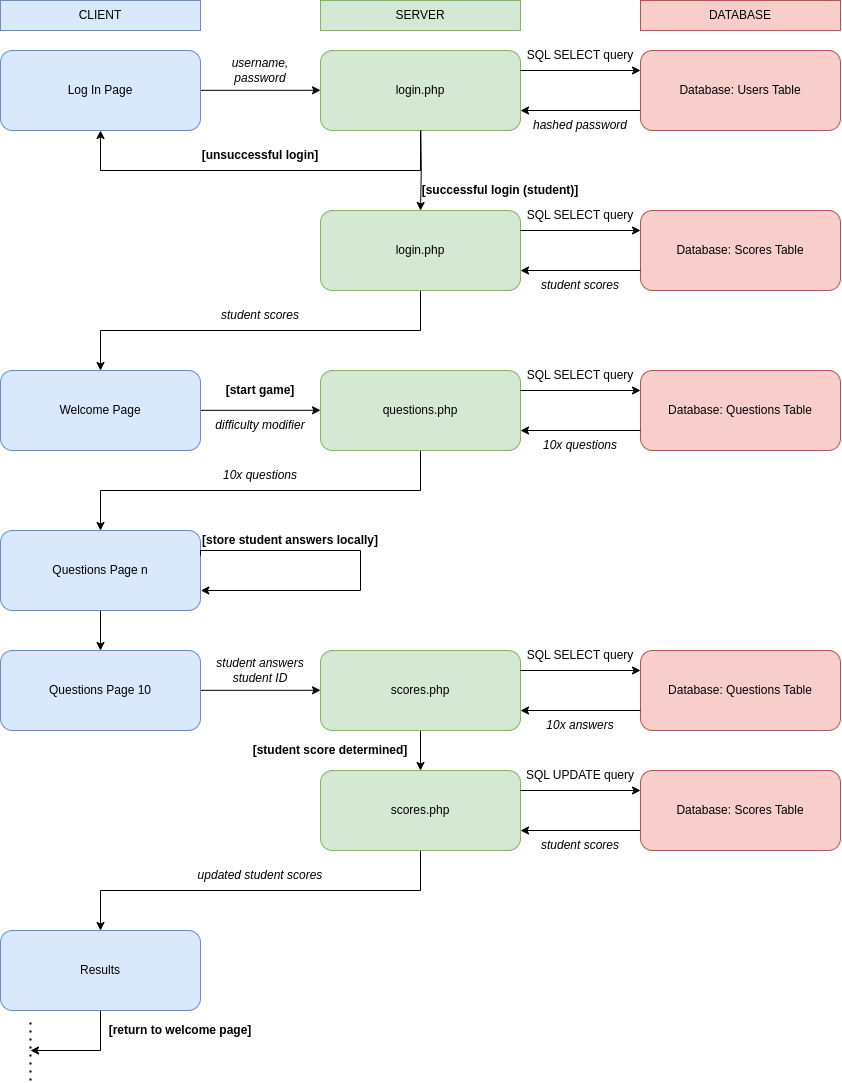
For each of the tables in the database, the following functionality is provided:

Clicking a button corresponding to a table in the database triggers the server to retrieve all records of the particular table. These are sent to the client as Json, the proposed user interface is rendered to the screen.

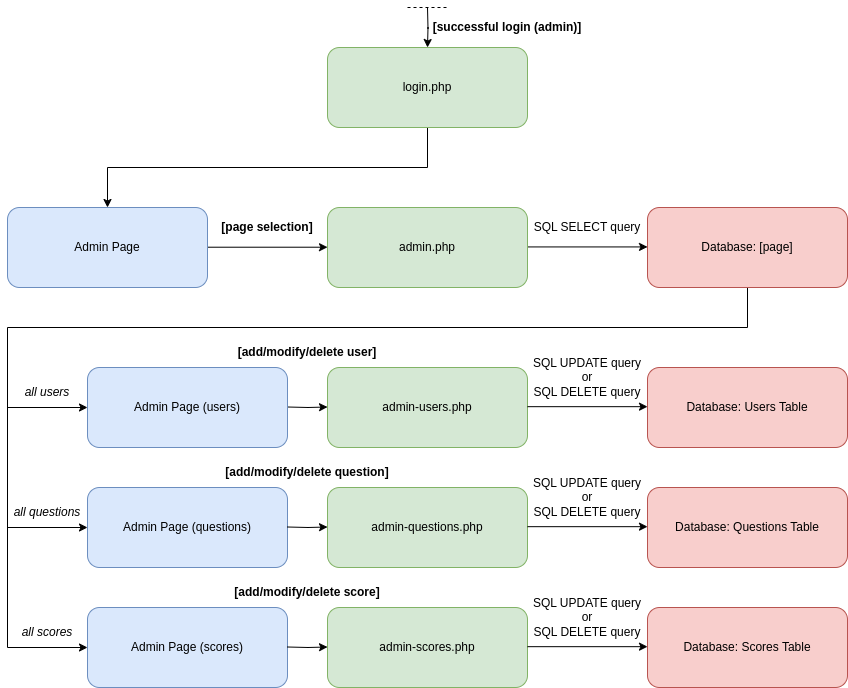
1. Filling in the input fields in the “add item” section and
2. Attempting to submit the data will cause client-side validation, and if successful, the data will be passed to the server which sanitises it.   
   *An appropriate SQL update query is performed, and a success (or failure) message in the form of Json is sent to the client based on the result of the procedure.*
3. Clicking the button to modify the item will provide an interface similar to the add record section, however the fields will be rendered containing the existing information. The user may make changes and submit the data, in which case a procedure similar to add item is performed.
4. Clicking the button to delete a record on the database will send the unique identifier for the record to the server, and a SQL delete query will remove it from the appropriate table.

**4.2.1 Control Flow Diagram: Student**

*Intro of the content – give the definition*



**4.2.2 Control Flow Diagram: Admin**



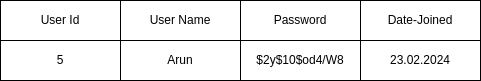
**4.3 Databases**

A detailed database system with a list of associated tables is necessary to store, manage, and manipulate the essential data on which the whole quiz system operates on, such as the user data, quiz questions, and recorded scores.

**4.3.1 Users Table**

Stores a username, hashed password date created for each user of the application.

Each record has a user ID which uniquely identifies that user.

**Fig X: DB schema and an example record of USERS table (User Id Primary key , User Name , Password , Date-Joined )**

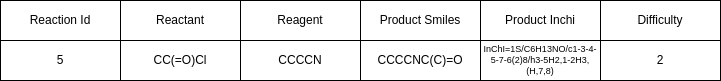
**4.3.2 Questions Tables**

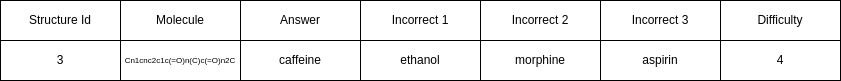
Each type of question will require its own table in the database. Each will contain information about the question to be asked, the correct answer, and a difficulty rating. Multiple-choice questions will include pre-chosen decoy answers. Each question will contain a unique ID to identify the particular question on the particular table.

Structural information about chemicals will be stored in two established formats:

SMILES (simplified molecular-input line-entry system): a way to represent chemicals in string format. For example, sodium chloride could be represented as "[Na+].[Cl-]". It has the ability to be transformed into a SVG (image format) on the client, using a library called RDKit.js. (REF: https://www.rdkitjs.com/)

InChI (International Chemical Identifier): a more recent method than SMILES to represent chemicals in string format. Unlike SMILES, chemicals in the InChI format are represented in a completely unambiguous manner. For example, sodium chloride is "InChI=1S/ClH.Na/h1H;/q;+1/p-1" in the InChI format, whereas both "[Na+].[Cl-]" and "[Cl-].[Na+]" are valid SMILES. A user's chemical structure input will therefore need to be converted to its InChI to compare with the stored InChI string answer. RDKit.js unfortunately does not have the ability to directly convert InChI strings into svgs (or into SMILES), so both formats are required.

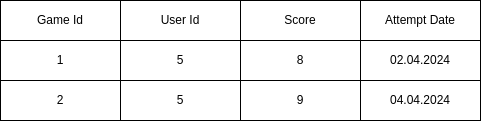
**Fix X: DB schema and an example record for REACTIONS table (Reaction Id Primary key, reactant, reagent, productSmile, productInchi, catalyst, solvent, temperature, time, difficulty)**

**Fig X: DB schema and an example record for STRUCTURE table (Structure Id Primary key, molecule, answer, wrong1, wrong2, wrong3, difficulty)**

4.3.3 Scores Table

Contains details about the scores a specific user makes for each of their quiz attempts – the user Id corresponding to an entry in the Users table, the total score and the attempt date.

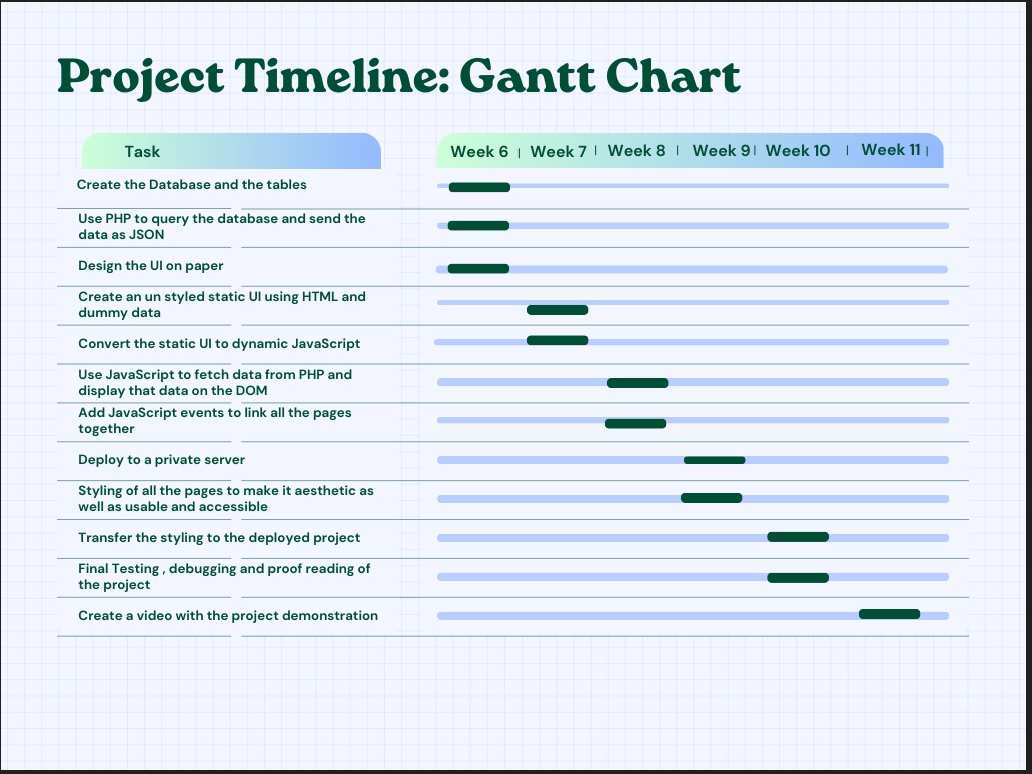
Every record has a Game Id to uniquely identify each quiz attempt.

**Fix X: DB schema and an example record for SCORES table (Game Id Primary key, User Id Foreign key, Score, Attempt Date)**

**5.Timeline for the project Execution:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Tasks | Start Date | End Date | Week no |
| Task 1 | Create the Database and the tables | 08.04.2024 | 14.04.2024 | 6 |
| Task 2 | Use PHP to query the database and send the data as JSON | 08.04.2024 | 14.04.2024 | 6 |
| Task 3 | Design the UI on paper | 08.04.2024 | 14.04.2024 | 6 |
| Task 4 | Create an un styled static UI using HTML and dummy data | 15.04.2024 | 21.04.2024 | 7 |
| Task 5 | Convert the static UI to dynamic JavaScript | 15.04.2024 | 21.04.2024 | 7 |
| Task 6 | Use JavaScript to fetch data from PHP and display that data on the DOM | 22.04.2024 | 28.04.2024 | 8 |
| Task 7 | Add JavaScript events to link all the pages together | 22.04.2024 | 28.04.2024 | 8 |
| Task 8 | Deploy to a private server | 29.04.2024 | 05.05.2024 | 9 |
| Task 9 | Styling of all the pages to make it aesthetic as well as usable and accessible | 29.04.2024 | 05.05.2024 | 9 |
| Task 10 | Transfer the styling to the deployed project | 06.05.2024 | 12.05.2024 | 10 |
| Task 11 | Final Testing, debugging and proof reading of the project | 06.05.2024 | 12.05.2024 | 10 |
| Task 12 | Create a video with the project demonstration | 13.05.2024 | 17.05.2024 | 11 |

**Ghantt chart laying out the project execution plan:**

****

**6.Risk Management**

The impact and likelihood of the proposed risks have been rated on a low-medium-high scale.

**6.1 General Risks:**

These risks outline generic concerns that should be addressed when developing the project.

**6.1.1 Data Integrity**

Impact: high

Likelihood: low

It's crucial to ensure that the user’s input is inspected before it is sent to the database. This can be achieved though client-side and server-side validation, as well as server-side sanitisation. Through these methods, inappropriate data processing or the storing of invalid data will be minimised.

**6.1.2 Performance**

Impact: medium

Likelihood: medium

The program should be written in such a way that users are not turned off by sluggish or inconsistence performance. It is is essential to store data in the database in a deliberate and meaningful way in order to ensure that the database queries perform in an optimized and performant manner. Since network requests necessarily introduce latency to the application, and the number of requests should be minimised.

**6.1.3 Cross-Platform Integration**

Impact: low

Likelihood: medium

The user interface should look consistent across a selection of the most commonly used browsers and devices, and the program should continue to perform as expected. Testing should therefore be performed on a variety of browsers and devices.

**6.1.4 Scalability**

Impact: low

Likelihood: low

Scalability should be kept in mind when designing the database. Selecting a database management system which is able to accommodate large numbers of concurrent users is essential for a web-based application. It should also be able to store and retrieve increasing numbers of records with only negligible drops in efficiency.

**6.1.5 Usability**

Impact: low

Likelihood: medium

The user interface to the application should be created with accessiliby standards in mind, which would ensure that users with disabilities are not prevented or disadvantaged when accessing the program's features.

**6.1.6 Deployment**

Impact: high

Likelihood: low

Seeing as the development environment is different to the production environment, it is essential to deploy the project as soon as a minimum viable product has been achieved. This is to ensure that any bugs or errors resulting from the different environment are dealt with as soon as possible.

**6.1.7 Data Loss**

Impact: high

Likelihood: low

It is necessary to often back up data for developing software, to prevent the loss of work due to hardware failure. The using of cloud-based services such as GitHub or Office365 help to mitigate this risk, as that is key to their design.

**6.2 Group Risks**

These risks are specific to our development team

**6.2.1 Knowledge Gaps**

Impact: medium

Likelihood: high

Different members of the team have different programming skills. The situation may arise where members of the team are finding an aspect of their assignment to be too difficult to perform in the proposed timeframe. In this scenario, the member of the team which has the most experience in the problem will provide assistance.

**6.2.2 Group Member Exit**

Impact: low

Likelihood: low

In the case that a member of the team is no longer able to perform their role, their tasks will be assigned to either the member of the team that nominates to do it, or split among the remaining members.

**6.2.3 Programming Language Fallbacks**

Impact: low

Likelihood: medium

Frontend: if the development of a user interface with vanilla JavaScript is proving to be too challenging, the more developer-friendly Svelte framework may be incorporated.

Backend/Database: if we are not able to use PHP to perform the required tasks, we could switch to a JavaScript-based library such as Express.js, or in an extreme case, incorporate a Backend-as-a-Service (BaaS) such as AWS or Supabase.

**7.Summary**

The proposed app is an innovative and genuine attempt to change improve chemistry education by utilizing the interactive advancements available in the digital age. The app seeks to empowers students to improve their understanding of chemical science by offering an engaging, hands-on learning platform, helping them to grow their critical thinking capabilities as well as developing a greater fascination for this subject.

Even though this is our core design concept and plan, Our Agile methodological

approach may cause us to make a few changes and updates in some areas where there might be scope for further improvement.

We thoroughly believe that we have the ability to transform this plan into a working project, and look forward to demonstrating its functionality to Instatute in due time.

**8.References**

1. Full, M. (2020). *How the Internet Works and the Web Development Process*. Independently Published.
2. Andrew, R., Ullman, C., & Waters, C. (2003). *Fundamental Web Design and Development Skills*. Glasshouse.
3. Simpson, J. (2023). *How JavaScript Works*. Apress.
4. Nixon, R. (2021). *Learning PHP, MySQL, and JavaScript* at “O’Reilly Media, Inc.”
5. Prettyman, S. (2020). *Learn PHP 8: using MySQL, JavaScript, CSS3, and HTML5*. Apress Media, Llc.
6. Smith, B. (2015). *Beginning JSON: [learn the preferred data format of the web]*. Apress.
7. Bassett, L. (2015). *Introduction to JavaScript Object Notation*, “O’Reilly Media, Inc.”
8. Delisle, M. (2006). *Creating Your MySQL Database*. Packt Publishing Ltd.
9. Gehani, N. (2011). *The Database Application Book Using the MySQL Database System*.
10. Davies, J. A. (2011). *Implementing SSL/TLS using cryptography and PKI*. Wiley; Chichester.
11. B. Bienfait and P. Ertl, [**JSME: a free molecule editor in JavaScript**](http://www.jcheminf.com/content/5/1/24), J. Cheminformatics 5:24 (2013)