# Group 7 Project Document

## INTRODUCTION

The objective of the Group 7 Project is to create a web application that encourages children to undertake responsibilities by gamifying task completion through a points-based reward system. The system allows an adult to set tasks for a child, mark them as completed, and then reward the child based on a points system. The child is then able to redeem rewards, based on the number of points they have accumulated. Our target audience is parents (responsible adult) and their children.

While our team initially conceptualised the web application under the moniker “Child Reward System” our collaborative journey and deeper understanding of its essence led us to a more evocative name: "Reward Quest." This new title not only encapsulates the gamified experience we aim to deliver but is also crafted to be catchy and resonate with our target audience. We believe "Reward Quest" aptly captures the adventurous journey of task completion and reward redemption, enticing parents and guardians to adopt our system in their quest to motivate their children.

### Roadmap of the report:

## BACKGROUND

### Our Vision

Our aspiration in developing "Reward Quest" was to offer more than just a utility - it was to blend functionality with enjoyment, crafting an experience that is both beneficial and fun.

“Reward Quest” is a web-based application meticulously designed to foster responsible behaviour and encourage task completion in children. Recognising the struggles guardians often face in motivating children to undertake chores, our solution seamlessly merges task management with a compelling incentive-based reward system.

### Accessing the Application

To access “Reward Quest”, launch the **app.py file** through PyCharm. Once it's running, simply click on, or navigate to, **http://127.0.0.1:3000** down in your terminal. This URL redirects you to the application's homepage, presenting a suite of options:

- Account Management:
    - Sign Up: Register a new account.
    - Log In: Access the application (for registered users).
    - Log Out: Securely exit your session.
- Task Management:

    - Create a Task: Define a new task for the child.
    - Mark a Task as Completed: Confirm the completion of a task.
    - Display Tasks assigned created.
    - View Completed Tasks: Review the tasks the child has finished.
- Reward System:
    - Create a Reward: Introduce a new reward for the child to aim for.
    - Display Available Rewards: View the range of rewards awaiting redemption.
    - Redeem a Reward: Once a Child has enough points, a Reward can be redeemed and the system will be updated.
- Insights:
    - View a Selection of Charts: Analyse the child's progress and engagement through information displayed on various charts.

## SPECIFICATIONS AND DESIGN

| **Functional Requirements:** | **Non-functional Requirements:** |
|---|---|
| Allow adults to add, modify, or delete tasks. | Safe and secure login. |
| Assign specific points to tasks. | Intuitive User Interface. |
| Update child's points in real time after task completion. | Real-time data processing. |
| Provide a reward redemption system. | Secure storage of user data. |
| Generate reports to show a child's progress. | Scalability to accommodate more users. |

## DESIGN AND ARCHITECTURE

The architecture of our project is designed to ensure secure access and efficient data management for users. Main areas of focus include:

1. **Clients:** represented by the laptop, users will access our application through their browsers. Considering our target audience, this would most likely be parents setting up accounts for their child(ren).
2. **Server:** symbolised by the server icon and yellow background, the server receives incoming requests from the user, processes them, and then sends responses back.
3. **API/modules:** within the server section, the top **'auth/JWS'** module handles user authentication and security through JSON Web Tokens so that authorised users can access any page where they have appropriate permission. The **'points'** module manages user points, rewards, and family members to store/retrieve the correct data. Public/private functions distinguish what is accessible to all versus pages that are only accessible to those with necessary permissions.
4. **Database:** the arrows connecting the modules to the database represent the connection between the application and the database that underlies the project. The **'users'** table would hold relevant user account information while the other tables would store points data and family information, along with any further information to enhance the usability of the project.

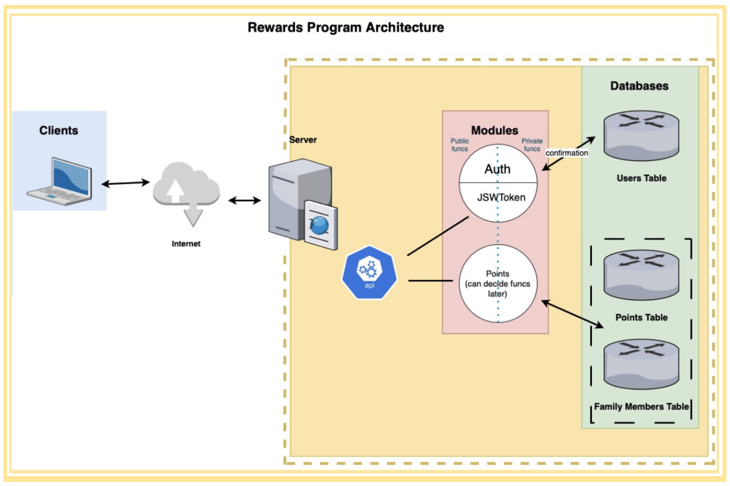

*Diagram of our project's architecture*

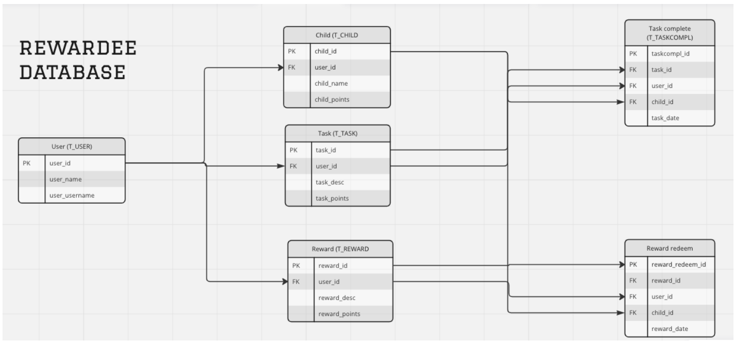

*Diagram of our Rewardee database*

This architecture focuses on modular design that allows the project to be broken into distinct parts that can be focused on when designating project tasks, as each module has a specific functionality. It also allows for scalability, as further modules could be added to add further functions to the application. Security measures are also implemented within the architecture to ensure that only authorised users can access certain aspects of the project, such as specific pages, and creates a seamless overall experience when navigating the application. Creating private functions also creates potential for personalised interactions based on the user's profile and permissions. Finally, by separating the project's logic into modules and interacting with the database, performance is optimised and ensures there is no redundant data storage.

## IMPLEMENTATION AND EXECUTION

Our development is based on regular brainstorming sessions during the course and outside the course, ideas sharing, and work distribution. We used Trello for backlog maintenance, prioritising, and tracking tasks. Mural was initially used to collaborate and brainstorm our project criteria, and we then used Figma to design the Webpages. Roles have been divided considering team member strengths and interests. Communication happens mainly through team meetings and Slack. We use GitHub as our source control where we create pull requests and ask other team members to peer review and approve them.

| **Roles** | **Members** |
|---|---|
| Frontend Team | Anneka, Sarah, Katrina |

| | |
|---|---|
| Backend Team | Elsa, Katrina, Klaudia |
| APIs Team (incl. authorisation) | Katrina, Klaudia, Elsa |
| Database Management Team | Katrina |

**Tools, Libraries & Modules:**

| **Tools, Libraries & Modules** | |
|---|---|
| Auth0 - our solution to add authentication and authorisation to our application | Version Control: Git, with repositories hosted on GitHub. |
| Web Development: CSS, HTML | Chart Visualisation: quickchart.io API |
| **-**Database Management: MySQLWorkbench | Flask, dotenv, AuthLib, functools, os, urllib, requests, HTTP library, unittest, mysql, requests-mock, DateTime library |

**Implementation process:**

- **GitHub Version Control:** We faced initial challenges in synchronising code via GitHub. Through shared best practices, visual aids, and screen-sharing, we improved our coordination.
- **Time Management & Team Dynamics**: Task allocation was time-consuming, pushing us to work intensively in the final weeks. Using Trello for tasks and Slack for updates, we managed despite varied personal commitments. At times, decisions were taken with only partial team presence.
- **MVP Development:** We prioritised the MVP's core features, placing secondary functions in the backlog. The final product surpassed our MVP expectations, featuring unanticipated additions like Authorisation and a polished user interface. With more time, further improvements are possible.
- **API Choices:** Initially eyeing the Amazon API, we found its integration challenging. We opted for quickchart.io, an API for charting. Adapting live data to this API presented challenges, especially when synchronising datasets for children.
- **Feature Refinement:** Some features evolved or were replaced based on feasibility and practicality. Enhancements like additional buttons improved webpage UX. We also introduced a login to streamline task and reward management for users.

**Agile development:**

- **Agile Ceremonies:** We streamlined our approach by prioritising coding, testing, and documentation over traditional ceremonies like daily stand-ups and sprint planning. While formal sprint reviews and backlog refinement weren't consistently done, we frequently reviewed and showcased our work. Sprint retrospectives were bypassed due to time constraints but remain a consideration for future reflection. Updates and blockers were communicated through Slack and impromptu Zoom sessions.
- **Pair Programming:** This agile technique was invaluable during our development phase, particularly when crafting the HTML and CSS pages. Collaborative coding not only facilitated the exchange of ideas but also expedited the process. Kanban Board: Our team utilised Trello, a Kanban-style board, to oversee and manage our work. Given our project's dynamics, we felt that a Kanban board was more suited to our needs than a Scrum board.

- **Acceptance Criteria:** While our primary focus was to progress with the coding, we also endeavoured to incorporate acceptance criteria whenever feasible. This ensured clarity regarding when a task could be deemed complete.
- **Refactoring & Code Reviews:** Throughout the project, we consistently refactored our code, aiming for optimisation and alignment with our MVP. We conducted code reviews via pull requests on GitHub and ensured that changes to the code were of a good standard before merging and that all conflicts were resolved. We often resolved these as a group in meetings. We shared screenshots and code files on Slack to share ideas, fix issues and ensure alignment.
- **Iterative Approach:** Our development strategy was inherently iterative. Beginning with a basic version of the website, we continuously enhanced it, refining the code until we achieved a design and functionality that went beyond our MVP.

## Implementation challenges:

- **Design and HTML**
    - Accessibility: Ensuring navigation within and between pages can accommodate as many users as possible - this would include creating pages with proper contrast and easy keyboard navigation in order to create a fully inclusive environment. The page has been designed with pastel colours in the background so that the writing can be easily viewed by the user. Also, navigation header tabs are highlighted whenever the mouse hovers over them, making them easier to navigate. Charts are displayed one at a time and are clearly labelled, which makes it easier for users to understand how information is being organised visually.
    - **Suggested improvements:**
        - Responsive design: the project currently focuses only on a webpage; however, if the project is scaled up to include various other devices, such as a phone screen or larger monitor, it would be important to maintain accessibility across all platforms, which would include changing font sizes and page layouts to suit the viewports.
        - Third-party APIs: as some APIs have their own designs, it's possible that those designs have been implemented without accessibility in mind.
- **Backend**
    - API complexities stem from the amount of external documentation and parameters that need to be considered; it has to be set up correctly in order to seamlessly integrate into the program. When building the program, we also need to consider the rate usage limitation of an API to avoid any disruptions when trying to use the service. Also, every API has its own way of outputting an error response; if we aren't familiar with how to handle these error messages, it would take longer to troubleshoot and progress tasks.
    - **Suggested improvements:**
        - Security audits: while security measures and reviews have been set up at a basic level with authorization and secured endpoints, regular code reviews would ensure more rigorous protection, which could be done through using tools like SonarQube (a tool that continuously inspects the quality of code and has the ability to detect bugs).
- **Database**
    - Data consistency and integrity can become a challenge for implementation, especially throughout the duration of building the project. As elements of the project change, aspects like primary and foreign keys and unique constraints have to be monitored/updated to ensure data consistency. Implementation issues include multiple users modifying data at the same time, which could leave to invalid or inaccurate data being recorded.
    - **Suggested improvements:**

        - Caching: caching mechanisms could be added to store data in memory, which would reduce load on the database.
        - Query optimization: database queries could be reviewed consistently to ensure the time it takes to execute these queries is minimal, thus using less resources.
        - Cloud services: depending on how large the project becomes, we could consider cloud-based databases that can automatically scale alongside the web application, while also providing tools that allow for ease of management and monitoring.

## TESTING AND EVALUATION

### App.py:

The main strategy implemented within this file involves manual testing and error handling and exception management, where different parts of the project are tested for functionality and security.

Manual testing was done by amending the code and manually navigating the different endpoints and use cases of the application to check whether expected outcomes are met. This was done by continuously checking that login/logout/signup functions would redirect to the correct page every time the code was amended, as well as ensuring that protected pages would redirect the user to the login/signup page if they weren’t an authorised user.

The try/except blocks prevent the application from crashing if an unexpected error occurs while the program is running. As an example, this would prevent incorrect token formats from parsing successfully while also resulting in an elegant error message that clearly informs the user of the specific issue.

The **potential limitations** include:

- Session management: the project currently relies on session data to store a user’s information. While this works for a small-scale project, an increase in users may create issues.
- Security issues: token-based authentication is being handled by Auth0 but further security measures may be implemented to ensure that token leakage and unauthorised access are avoided.

Considering the above, **suggested improvements** include:

- Testing for usability: allowing users to test out the application and provide feedback would give us deeper insight into any usability issues, such as broken links or unprotected endpoints that have gone unnoticed.
- Edge cases: testing edge cases, such as incorrect tokens or invalid input, would allow us to catch any atypical program application uses so that any potential issue can be handled swiftly.

## CONCLUSION & KEY TAKEAWAYS

Throughout the development of 'Reward Quest,' our primary aim was to digitise and gamify children's routine tasks, thereby fostering a more interactive engagement between parents and children. Despite encountering several challenges, the finalised application is a reflection of our commitment and systematic approach.

### KEY TAKEAWAYS:

- **Naming Matters:** Our initial inclination was "Child Reward System." However, after deliberative brainstorming, we settled on "Reward Quest." It's not only catchy but encapsulates the app's adventurous spirit.

- **Effective Teamwork:** Our toolkit comprised Trello, Slack, and GitHub. The team, boasting a diverse skill set, often showcased individual brilliance. Yet, synchronisation at times posed hurdles. Regular check-ins and the occasional caffeine boost saw us through.
- **Adaptability is Key:** Initially, we aimed to integrate the Amazon API. However, circumstances led us to pivot towards quickchart.io – a move that seamlessly aligned with our vision.
- **MVP's Core Essence:** With time constraints looming, it was paramount to distil the app's core features and postpone the add-ons. This strategy ensured we remained focused and didn’t overextend (too much!).
- **Inclusive Design:** Beyond a functional interface, our design ethos was centred on accessibility. Although strides were made with user-friendly colour palettes and layout choices, there's potential for further improvements, especially in areas like keyboard accessibility and cross-device compatibility.