Software Requirements Specification

for

ChoreTracker2

Version 1.0

Prepared by Billy Andrews, John Hurnyak, & Hunter Renard

University of Mary Washington

September 20, 2017

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to describe the ChoreTracker2 software. This document contains the functional, behavioural, and non-functional requirements of the project and also contains the guidelines for system engineers and designers to start working the project.

1.2 Scope

The ChoreTracker2 is a chore completion tracking system intended for parents to keep track of their children's daily contribution to the household.

The project was designed specifically for parents of adolescent chore-tasked children. The product will work as a complete user interface for updating and tracking daily chores marked for completion. The ChoreTracker2 application is a web-based application for interoperability across multiple devices. ChoreTracker2 can be used from any web browser and will be responsive for use on mobile platforms. The requirements in this document have been discussed with and requested by the client, Dr. Anewalt.

1.3 References

https://web.cs.dal.ca/~hawkey/3130/srs_template-ieee.doc was used to reference the standards of procedures of writing requirements documents and was used as a template.

1.4 Overview of the remainder of the document

The entirety of this document is split among six sections:
Section 1 is the introductory documentation. This section displays a brief description and explains the scope of the project and also contains the references used within the document.
Section 2 is the overall description of the product itself. This section elaborates further into how the product will be used by the users and also touches on general constraints.
Section 3 delves into the requirements of the ChoreTracker2. This section answers the specifics and provides a framework for the project.
Section 4 discusses all of the non-requirements,in other words, the requirements that are explicitly not desired.
Section 5 talks about the assumptions that are to be made while designing the systems.
Section 6 of this document is the appendices, which just defines any field related or ambiguous terms and names the authors of this document.

2. General Description

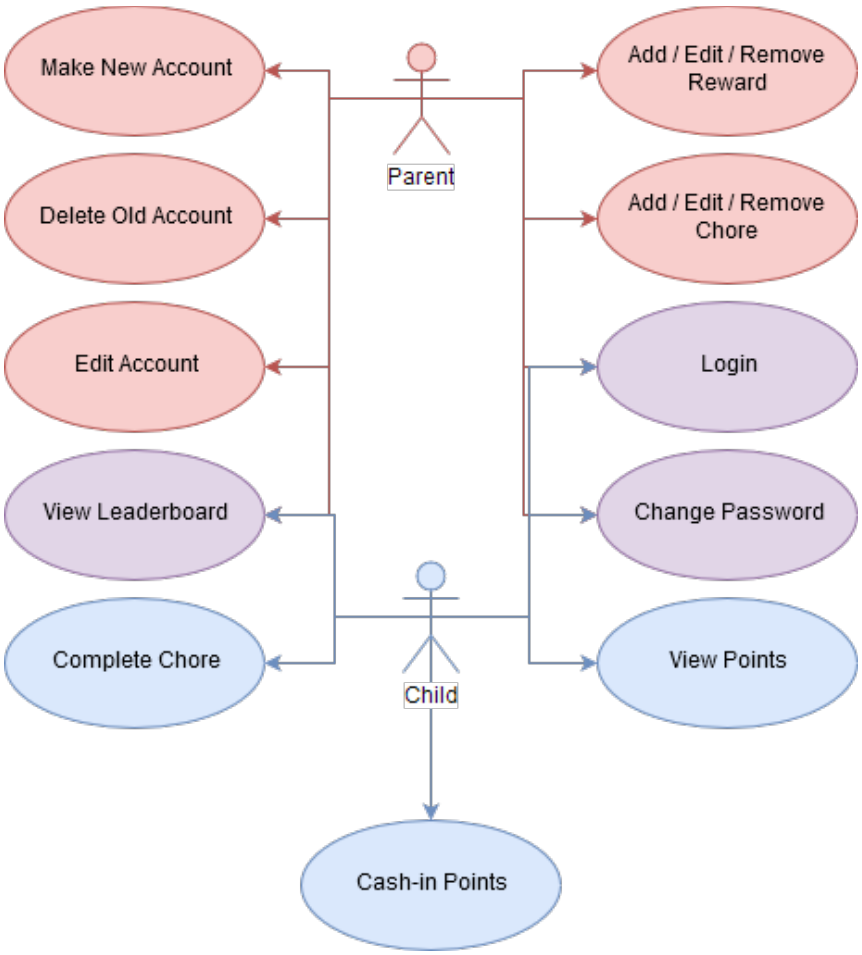
2.1 Product functions

The ChoreTracker2 software will retain a database of completion dates along with the chore itself that was completed and how many points the child earned for completing that particular task. The software will also

retain the amount of points the child has saved and can be traded for various available awards within the program's electronic reward store.

The Child user allows functions of logging in, completing a chore, and trading their points in for specific rewards decided upon by the Parent account holders within the scope of that family.

The Parent user allows functions of logging in, creating new chores and destroying old chores, as well as also creating and destroying rewards of any type.



2.2 User characteristics

There will be two types of accounts: **Parent** & **Child.**

Parent accounts are administrative accounts that have access to creating and destroying child accounts. They also have full access to all options within the scope of the software, like password management and chore creation with assignment of rewards for specific chores.

Child accounts do not have full access. Children will have the ability to login, submit completion requests, and cash-in their points for rewards. The child accounts are not considered trusted account types to eliminate the possibility of nefarious acts unbeknowingly to the parent.

2.3 General constraints

The client requested that the software be either a web-based app or an iOS app. Due to the unavailability of iOS devices for some potential users, ChoreTracker2 will be built as a web application. This will also ensure interoperability by enabling non-iOS users to have access to the application.

3. Requirements

3.1 General Requirements

3.1.1. The System MUST have user accounts with parent or child user privileges.
3.1.2. The System SHOULD send emails to a child user's parent user when the child user claims a reward.
3.1.3. The System MUST allow a parent user to create chores
3.1.4. The System MUST allow a parent user to create rewards
3.1.5. The System MUST allow a parent user to create a child User.
3.1.6. The System MUST allow a parent user to change a child user's personal data.
3.1.7. The System MUST allow a parent user to change a child user's login information.
3.1.8. The System MUST allow a parent user to remove a reward.
3.1.9. The System MUST allow a parent user to deduct points from a child user.
3.1.10. The System MUST allow a parent user to remove a chore
3.1.11. The System MUST allow a parent to remove a child user
3.1.12. The System MUST allow a parent user to login.
3.1.13. The System MUST allow a child user to login.
3.1.14. The System MUST allow a parent user to delete their account.
3.1.15. The System MUST allow a parent user to edit their account
3.1.16. The System MUST require an email is verified before the user may login.
3.1.17. The System MAY require reward points to be within a certain range (EX: between -9999 and 9999).
3.1.18. The System MUST allow a parent to manually reward / deduct points for a child user.
3.1.19. The System MUST allow a child user to view their list of chores.
3.1.20. The System MUST allow a child user to view their list of rewards.
3.1.21. The System MUST allow a child user to mark a chore as completed.
3.1.22. The System MUST allow a child user to cash in points for a reward.

3.1.23. The System MUST allow a parent user to deduct points from a Child User.
3.1.24. The System MUST allow a parent to revoke a child user's rewards.
3.1.25. The System SHOULD allow a parent to set reward limits (such as only cashing in points for a specific reward _ number of times).
3.1.26. The System MUST allow parent users to edit chores.
3.1.27. The System MUST allow parent users to edit rewards.
3.1.28. The System SHOULD allow child users to request points.
3.1.29. The System MUST be accessible from a web browser.
3.1.30. The System MAY be accessible by using an iOS application.
3.1.31. The System MAY be accessible by using an Android application.
3.1.32. The System MUST utilize a database system, such as Postgresql.
3.1.33. The System MAY allow a web browser to cache information in order to speed up logging in to the system User Interface.

3.2 Capacity requirements

3.2.1. The System SHALL have at least 1GB (gigabyte) memory.
3.2.2. The System SHALL be enabled to moderate traffic simultaneously.

3.3 User interface requirements

3.3.1. The System MUST not allow child users to view the Parent User Interface.
3.3.2. The System MAY allow Parent users to view the Child User interface.
3.3.3. The System MAY allow all users to view terms and conditions.
3.3.4. The System MAY allow all users to view a project team page.
3.3.5. The System MUST allow a child user to view their points.
3.3.6. The System MUST allow a parent user to view a child user's points.

3.4 Network requirements

3.4.1. The System MUST have an active internet connection.
3.4.2. The System MUST be able to handle multiple connections simultaneously.
3.4.3. The System SHALL be hosted on a cloud service, such as Google Cloud or AWS.

3.5 Error handling requirements

3.5.1. The System MUST alert the current user to any error which might interfere with the completion of an action.

3.6 Security requirements

3.6.1. The System MUST utilize a username/password scheme stored in the postgresql database.
3.6.2. The System MUST only allow authorized users to use "access controlled" web resources.
3.6.3. The System MUST allow parent accounts to access and edit passwords of child accounts.
3.6.4. The System MUST encrypt both parent and child passwords.

3.7 Input / Output requirements

3.7.1. The System MUST accept a completed chore as input.

3.7.2. The System MUST accept a username / password combination as input.
3.7.3. The System MUST require a username, password, full name and email for a parent account to be created.
3.7.4. The System MUST require a username, password, full name, and email for a child account to be created.
3.7.5. The System MUST require all inputs to be a valid format for the input type (EX: an email being entered as username@example.com).
3.7.6. The System MUST display a list of chores available to the logged in Child.
3.7.7. The System MUST email the Parent User every time a Child user cashes-in their points.
3.7.8. The System MUST email the Parent User every time a Child user completes a chore.

3.8 Processing requirements

3.8.1. The System MUST process the total amount of points available to every Child account.
3.8.2. The System MUST provide Parent accounts with detailed statistics of every Child account within the scope of that family.

4. Non-Requirements

NR.1. The System MUST NOT send daily email summaries.
NR.2. The System MUST NOT run without first configuring a Parent account.
NR.3. The System MUST NOT allow a child user to create chores.
NR.4. The System MUST NOT allow a child user to create rewards
NR.5. The System MUST NOT allow a child user to create a child user.
NR.6. The System MUST NOT allow a parent user to create a parent user.
NR.7. The System MUST NOT allow a child user create a parent user.
NR.8. The System MUST NOT allow a child user to change their personal data.
NR.9. The System MUST NOT allow a child user to change their login data.
NR.10. The System MUST NOT allow a child user to change a parent user's personal data.
NR.11. The System MUST NOT allow a child user to change a parent user's login data.
NR.12. The System MUST NOT break in the instance of an error, rather redirect the user to an error-free environment.
NR.13. The System MUST NOT be vulnerable to SQL injection.
NR.14. The System MUST NOT allow multiple instances of one user account to access web resources simultaneously.

5. Assumptions

1. The system back-end is assumed to be written using Python with a flask template.
2. The system database is assumed to be PostgreSQL.
3. The web-host for the system is assumed to be Amazon Web Services.
4. The system front-end will be made responsive by implementing Bootstrap 4.

6. Appendices

6.1 Glossary

Interoperability

The ability of computer systems or software to exchange and make use of information.: "interoperability between devices made by different manufacturers".

Responsive

A responsive website automatically changes to fit the device that the user is viewing it on.

6.2 Authors

Billy Andrews: Contributed to the Introduction, General Description, Requirements, Appendices, and Assumptions.
John Hurnyak: Contributed to the Introduction, General Description, Requirements.
Hunter Renard: Contributed to the Introduction, General Description, Requirements, and Assumptions.