FamiOwl

Qichao Lin, Xinyu Meng, Yuchen Wei, Wendi Huang, Chengyu Liang

1. System Requirements

1.1 Functional Requirements

FAM-001 Parents can set the time limit of game time for each of their children on the parent portal.

FAM-002 Parent can access portal by mobile phone.

FAM-003 Once a child reached the time limit their parents set, the game would automatically exit.

FAM-004 Parents can see how much time in total their children have played for each game on the parent portal.

FAM-005 Children can download, install, and play games in one tap.

FAM-006 Each child has their own profiles.

FAM-007 Each child has their own game libraries.

FAM-008 Each game has their own game profile page.

FAM-009 The child client can record game time for each child.

FAM-010 Parents need an account to use the parent portal and the child client.

FAM-011 The modification of any time limit rule would apply the next time the corresponding game starts.

FAM-012 The data and profiles of children shall persist when the parent sign-in to the child client on a different machine.

FAM-013 Games in the library shall have a big square icon with their names under to make them easy to select.

FAM-015 Children and parents can search games based on tags.

FAM-016 Children and parents can search games based on names.

FAM-019 The child client shall be able to let parents sign-up.

FAM-020 The parent portal shall be able to let parents sign-up.

FAM-021 Elements, icons, UIs shall resize and adapt automatically when the parent is accessing the parent portal on their mobile devices.

FAM-022 Parents can also set an overall time limit for any game.

1.2 Non-Functional Requirements

FAM-101 Children must use a Windows or Mac computer to run the child client.

FAM-102 Communication between the child client and the server shall be protected under SSH Tunnel.

FAM-103 The whole development must be done within the semester

FAM-104 Parents must have a web browser on their computer or phone to access the parent portal.

FAM-105 The game in the library must run when the child client is running.

FAM-106 Games other than those published on the child client would not be constrained by the rule that the parent set.

FAM-107 Parents and children must use the Rutgers VPN to connect to the database and web server if they are off-campus.

FAM-108 Children's computer must have python installed.

FAM-109 The performance of the database and web server is poor as there's no budget for renting a better server.

2. Test Design

Test Case ID	*T01*
Purpose	Web Server Sign up testing
Pre-conditions	No testing data in the database before sign up.
Inputs/Test data	Username: tester Email: tester@gmail.com Password:tester
Expected Outputs	Sign up successful screen and new data is appeared in the database parent table
Post-conditions	New user data is appeared in the database parent table
Design Technique	*reviewing requirement.*

Test Case ID	*T02*
Purpose	Children Client Sign up testing
Pre-conditions	No testing data in the database before sign up in Children Client
Inputs/Test data	Username: test3 Email: test3@test.com Password: test

Expected Outputs	Sign up successful screen and new account show up in the database parent table
Post-conditions	New user data is appeared in the database parent table
Design Technique	*reviewing requirement*

Test Case ID	*T03*
Purpose	Web Server log in testing
Pre-conditions	Several user data existed in the database.
Inputs/Test data	correct data: Username: tester Email: tester@gmail.com Password:tester wrong data: Username: 12345 Email: 12345@gmail.com Password:123
Expected Outputs	log in successful screen with the correct data and log in failure with the wrong data
Post-conditions	No data changed
Design Technique	*equivalence class partitioning*

Test Case ID	*T04*
Purpose	Children Client log in testing
Pre-conditions	There are existing accounts in database
Inputs/Test data	Username: test3 Email: test3@test.com Password: test
Expected Outputs	Login successful screen and go to kids accounts selection page
Post-conditions	Parent account`s information can be shown correctly in main page
Design Technique	*equivalence class partitioning*

Test Case ID	*T05*
Purpose	Each child account has their own profile
Pre-conditions	There are unique data stored in database for each kid account
Inputs/Test data	switch between different children accounts in children selection page
Expected Outputs	the profile shows unique information for different kids
Post-conditions	Profile information for different children accounts can be shown correctly in main page
Design Technique	*reviewing requirement*

Test Case ID	*T06*
Purpose	Parents can set the time limit for their kids
Pre-conditions	There are one or more kids belong to this parent account
Inputs/Test data	valid time limitation value and invalid time limitation value
Expected Outputs	the time limit in database will change if the input is valid time value. warning will appear if the time limit is invalid (negative)
Post-conditions	check if new time limit appear in the databases
Design Technique	*equivalence class partitioning*

Test Case ID	*T07*
Purpose	Parents can access portal by their mobile devices
Pre-conditions	Web server is running on the VM
Inputs/Test data	input the web address into the browser of smart phone
Expected Outputs	loading the same web page as pc
Post-conditions	check if UI is the same as the pc browser
Design Technique	*reviewing requirement*

Test Case ID	*T08*
Purpose	Each child has their own game libraries
Pre-conditions	There`s a table in database for downloaded games by different kids so that every kid can have unique inventory.
Inputs/Test data	Select “kid1” kid account under “test@test.com” parent account and click on the inventory button.
Expected Outputs	the inventory shows unique information for kid1
Post-conditions	game information for different children accounts can be shown correctly in main page
Design Technique	*equivalence class partitioning*

Test Case ID	*T09*
Purpose	The child client can record game time
Pre-conditions	There`s time limits that are set up by the parents for their kids.
Inputs/Test data	log in “test@test.com” and select “kid1”, then test if the timer is showing the correct time limit and countdown successfully.
Expected Outputs	The timer is showing the correct time limit and countdown successfully.

Post-conditions	Check if time limit is updated after shut down client.
Design Technique	*equivalence class partitioning*

Test Case ID	*T10*
Purpose	Children and parents can download game from store into own inventory
Pre-conditions	There`s a store for user to view and download games.
Inputs/Test data	log in with "test@test.com" and kid account "kid1", click on "Snake" shown in the store
Expected Outputs	The Snake Game was downloaded and appeared in the inventory.
Post-conditions	Check if the game is downloaded successfully.
Design Technique	*reviewing requirement*

Test Case ID	*T11*
Purpose	Children and parents can search games based on names.
Pre-conditions	There`s a search bar for user to search games.
Inputs/Test data	search "Snake"
Expected Outputs	The Game "Snake" show up after searching
Post-conditions	Check if the Game shows up with all features(like description and runnable button)
Design Technique	*equivalence class partitioning*

Test Case ID	*T12*
Purpose	Children can play the games if the time limit is not reached.
Pre-conditions	The time limit is not reached, and the game is downloaded
Inputs/Test data	use "kid1" kid account under "test@test.com" to play Snake
Expected Outputs	The Snake game is runnable, and the user can successfully exit the game.
Post-conditions	Check if the Game runs with all features, and the timer keep working when the timer is working
Design Technique	*reviewing requirement*

Test Case ID	*T13*
Purpose	Parent can get child object with their user id
Pre-conditions	Parents has registered with their corresponding kids and logged in

Inputs/Test data	use test account for parent and the kids name
Expected Outputs	Check if the kids objects can be found by a parent test account with the user_id
Post-conditions	Check if the resulting kids object has the corresponding kid objects
Design Technique	*reviewing requirement*

Design as many test cases as needed to cover all functional and non-functional requirement listed in Section 1. Each test case should follow the above table format. If you have 30 test cases, you should have 30 tables.

3. Traceability

Test Case Number	List of the Requirements tested
T01	FAM-010, FAM-020
T02	FAM-019
T03	FAM-010
T04	FAM-012
T05	FAM-006
T06	FAM-001, FAM-022
T07	FAM-104, FAM-021
T07	FAM-002
T08	FAM-007
T09	FAM-009
T10	FAM-005
T11	FAM-015,FAM-016
T12	FAM-008,FAM-105,FAM-013
T13	FAM-001, FAM-012

Create a traceability table below. Make sure each requirement is tested by some test cases to show that the software product is meeting the requirements, including functional and non-functional.