

Final Report - FamiOwl

S-Team: Yuchen Wei, Wendi Huang, Chengyu Liang, Xinyu Meng, Qichao Lin

1. Implementation Report

Requirement	Implementation Plan	Progress	Comments
FAM-001	Parents can set the time limit of game time for each of their children on the parent portal.	100%	Done. Now parent can set the time limit in the my_children page.
FAM-002	Parents can set time limits for each game separately on the parent portal.	0%	The feature of setting time limit to each game is aborted due to time constraints.
FAM-003	Once a child reached the time limit their parents set, the game would automatically exit.	100%	
FAM-004	Parents can see how much time in total their children have played for each game on the parent portal.	100%	Real-time showing
FAM-005	Children can download, install, and play games in one tap.	80%	There is no installation required by this stage, and children need to add the game to inventory first to play it.
FAM-006	Each child has their own profiles.	100%	
FAM-007	Each child has their own game libraries.	0%	We think it would be strange if each child has to add their own games, so we decide that they all share the same game inventory that belongs to the parent.
FAM-008	Each game has their own game profile page.	100%	

Requirement	Implementation Plan	Progress	Comments
FAM-009	The child client can record game time for each child.	100%	
FAM-010	Parents need an account to use the parent portal and the child client.	100%	Done. Parents need to sign up first before using the parent web portal.
FAM-011	The modification of any time limit rule would apply the next time the corresponding game starts.	50%	The time limit would only apply the next time the client launches.
FAM-012	The data and profiles of children shall persist when the parent sign-in to the child client on a different machine.	100%	The communication between parents portal and child client is built successfully.
FAM-013	Games in the library shall have a big square icon with their names under to make them easy to select.	100%	
FAM-014	Games shall have tags for categorizing.	0%	We did not have enough time to modify our database structure to implement the feature.
FAM-015	Children and parents can search games based on tags.	0%	Same as FAM-015
FAM-016	Children and parents can search games based on names.	100%	
FAM-017	Children and parents can hit "Like" to a game.	100%	
FAM-018	The game profile page shall display the number of "Likes" a game has.	100%	

Requirement	Implementation Plan	Progress	Comments
FAM-019	The child client shall be able to let parents sign-up.	100%	
FAM-020	The parent portal shall be able to let parents sign-up.	100%	Done.
FAM-021	Elements, icons, UIs shall resize and adapt automatically when the parent is accessing the parent portal on their mobile devices.	100%	
FAM-022	Parents can also set a overall time limit for any game.	100%	Done. This function has been finished in the my_children page.
FAM-101	Children must use a Windows or Mac computer to run the child client.	100%	We managed to test the client on both Mac and Windows machine.
FAM-102	Communication between the child client and the server shall be protected under SSH Tunnel.	100%	
FAM-103	The whole development must be done within the semester.	100%	Done.
FAM-104	Parents must have a web browser on their computer or phone to access the parent portal.	100%	True. Now parents can access parent web portal through pc or smart phone.
FAM-105	The game in the library must run when the child client is running.	90%	We cannot guarantee the behaviour of this.
FAM-106	Games other than those published on the child	100%	

Requirement	Implementation Plan	Progress	Comments
	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.	100%	True.
FAM-108	Children's computer must have python installed.	80%	We do scan for the proper Python version on the system, but we cannot automatically install Python and other libraries.

2. Build and Run Instructions

Child Client:

1. Install Python 3.9.x or higher on the environment.
2. Create a Pyenv virtual environment.
3. Install required packages by running, `pip/pip3 install -r requirements.txt`, in the root directory of the project.
4. Run, `python/python3 main.py`, in the root directory of the project.

Web Server (Parent Portal)

1. Install Python/Python3 on the environment
2. Install required packages by running, `pip/pip3 install -r requirements.txt`, in the root directory of the project.
3. Run or `python/python3 manage.py runserver [ip_address:port_number]`


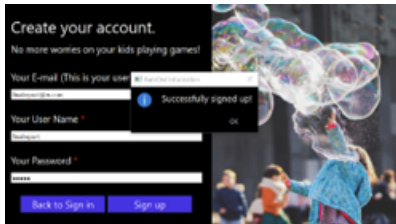


3. Bug Report


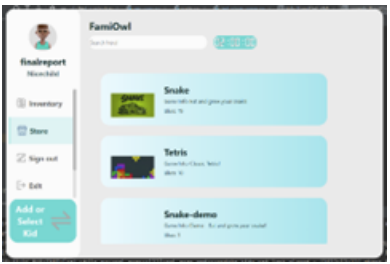
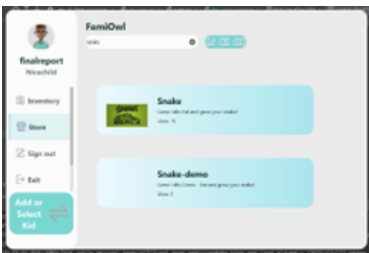
- After re-sign-in to the client, user can hit like to the games they liked before, and likes count will accumulate normally
- Newly published game will not appear to the game store immediately. It will appear the next time user sign-in to the client.
- Newly added kid is not selected automatically
- There should be parent's permission or password needed for adding or swaping a kid, signing-out and exiting.
- If the user force quit the program, any modifications made would not be saved.
- Games can be played without the control of the client using command lines.
- There exists limitation for negative number as inputting time limit but no maximum number limitation.


- The parent cannot see the game dictionary of kids.

4. Screen Shot

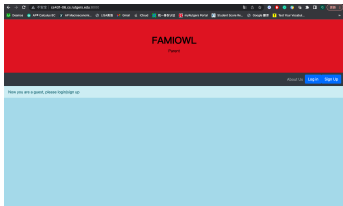
Child Client Screen shots:

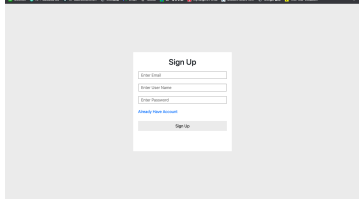
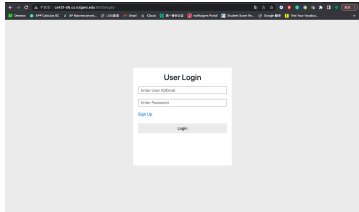
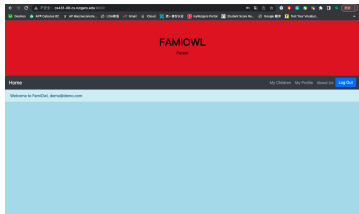
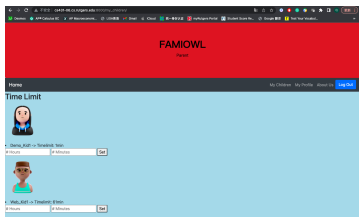
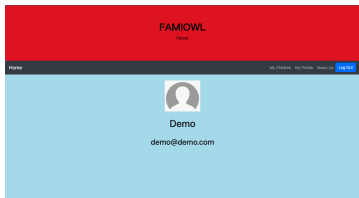
Name of the window	Screen shot	Description	Navigates to:
#1 Login window		Allow user to input their username.	1: Auto login: #6 child client home page 2: First time login: #3 Select child page 3: Sign-up: #2 sign-up window
#2 Sign-up window		Allow user to input sign up information	1: back to sign in: #1 login window 2: sign up: #1 login window
#3 Select child window		Allow user to select a child or add a new child	1: add a new child: #4 create child pop up 2: selected a child: #5 game inventory
#4 Create child pop up		Allow user to select icons and input name for new child	1: add child: #5 game inventory

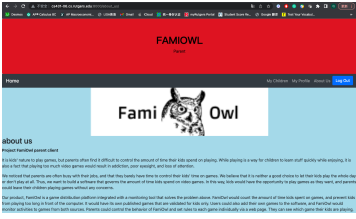

<p>#5 Child client home page (Game Inventory)</p>		<p>Allow user to start games that are in the inventory</p>	<p>1: start game: the game 2: search: #7 search results 3: store: #6 game store 4: Sign off: close, does not save login info, next time: #1 login window 5: exit: close, net time: #8 auto login</p>
<p>#6 Game store</p>		<p>Allow user to browse game and add game to inventory</p>	<p>1: add game: stay here, user will be able to see the game in inventory later 2: search: #7 search results 3: inventory: #5 game inventory 4: Sign off: close, does not save login info, next time: #1 login window 5: exit: close, net time: #8 auto login</p>
<p>#7 Search results</p>		<p>show search results</p>	<p>1: add game: stay here, user will be able to see the game in inventory later 2: start game: the game 3: store: #6 game store 4: search: #7 search results 5: inventory: #5 game inventory 6: Sign off: close, does not save login info, next time: #1 login window 7: exit: close, net time: #8 auto login</p>

#8 auto login		Auto login to the client with saved login info. Buttons will freeze, animation will keep moving.	will go to #5 game inventory directly
---------------	---	--	---------------------------------------

Parent Portal Screenshot:

Name of Page	Screenshot:	Description:	Navigates
#1. home_page		This is the home for the guest user. They can only see the about us page. People can use the function of parent portal only after they sign up their accounts.	#7: About_us page part 1, #8: About_us page part 2 #2. sign_up page #3. login page

#2. sign_up page		This is the Sign Up page for the new user. New user need to provide email, username and password.	#3. login page
#3. login page		This is Log in page for the users who have account in the database.	#2. sign_up page
#4: User_h ome page		After login, user can see the user_home page. Parent User can go to my_children page or my_profile page or About_us page.	#4: User_home page
#5: my_chil dren page		This is the my children page for the parent users. Parent users can see their current kids and the time limit for each kid. If parent users want to change the time limit, they can fill in the new time limit and click set button, and the new time limit will be set to the database.	#4: User_home page #5: my_children page #6: my_profile page #7: About_us page part 1 #8: About_us page part 2
#6: my_pro file page		This is my profile page, which contains username and email address of user.	#4: User_home page #5: my_children page #6: my_profile page #7: About_us page part 1 #8: About_us page part 2

#7: About_us page part 1		This is about us page, which introduce our team and the contact information of every members.	#4: User_home page #5: my_children page #6: my_profile page #7: About_us page part 1 #8: About_us page part 2
#8: About_us page part 2		Same as #7	#4: User_home page #5: my_children page #6: my_profile page #7: About_us page part 1 #8: About_us page part 2

5. Project Metric

Parent Portal:

- 911 Manually LOC + 25 External-Source LOC = 936 LOC in Total(html 462 LOC, py 474 LOC)
- 5 external Python packages: PyMySQL==1.0.2, cryptography==38.0.1, pycryptodomex==3.15.0, Django==4.1.2, sqlparse==0.4.3
- 23 source files, 7 classes - 30 methods in total:

aes_pass.py: 1 class, 7 methods

model.py: 4 class, 1 method

test_models.py: 1 class, 4 methods

view.py: 7 methods

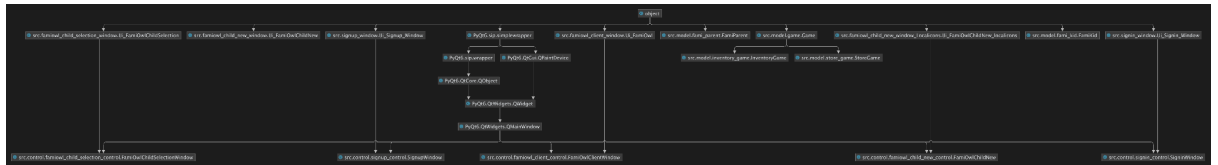
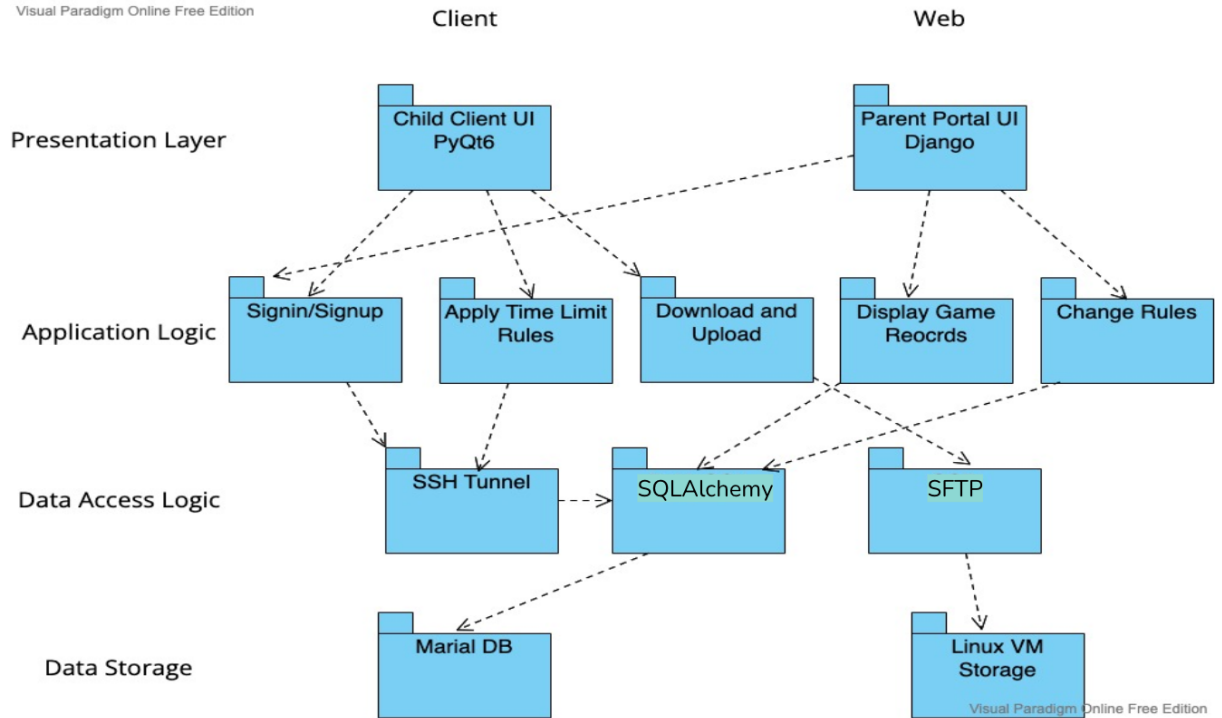
test_views.py: 1 class, 11 methods

Child Client:

- 2245 Manually LOC + 1237 Auto-Generated LOC + 303 External-Source LOC = 3785 LOC in Total
- 10 external Python packages: paramiko==2.11.0, psutil==5.9.2, pycopg2-binary==2.9.5, pycryptodomex==3.16.0, pygame==2.1.2, PyQt6==6.4.0, python_bcrypt==0.3.2, simplejson==3.18.0, SQLAlchemy==1.4.42, sshunnel==0.4.0
- 66 source files, 32 classes - 221 methods in total:

FtpInfo - 1, SftpUtils - 7, DbUtils - 4, SqlInfo - 2, SqlUtils - 6, ShellCmdInterface - 5, SshInfo - 2, SshUtils - 8, AESCipher - 7, WorkerSignals - 2, FamiOwlChildNew - 11, FamiOwlChildSelectionWindow - 11, FamiOwlClientWindow - 26, SigninWindow - 14, SignupWindow - 15, FamiKid - 15, FamiParent - 18, Game - 7, InventoryGame - 13, StoreGame - 10, Ui_FamiOwlChildNew - 2, Ui_FamiOwlChildNew_localicons - 2, Ui_FamiOwlChildSelection - 2, Ui_FamiOwl - 2, Ui_Signin_Window - 2, Ui_Signup_Window - 2, TestAesPass - 4, TestFamiKid - 2, TestFamiParent - 4, TestInventoryGame - 6, TestSshUtils - 4, TestStoreGame - 5

6. Software Structure and System Architecture



7. Summary

● Initial Plan vs. Final Product

Initial plan:

1. Have a child client and a parent portal
2. Child client should allow its user to browse games, download games.
3. Child client should shut down game according to the time set from the parent portal
4. Search game through child client
5. Tagging games in child client
6. Parent Portal display playtime
7. Parent Portal realtime monitor child client's activity

Removed during requirement engineering pace

8. Chatting function
9. Game publisher's client

Final Product:

1. Have a child client and a parent portal
2. Child client should allow its user to browse games, download games.
3. Child client should shut down game according to the time set from the parent portal
4. Search game through child client
5. ~~Tagging games in child client~~
6. Parent Portal display playtime **asynchronously**
7. ~~Parent Portal realtime monitor child client's activity~~
8. ~~Chatting function~~
9. ~~Game publisher's client~~

At the beginning of the project, we overestimated our ability to implement the program because we did not know the capabilities of our teammates. Also because we are unfamiliar with the tech stacks we were going to use, we underestimated the difficulty of implementing some of the requirements.

Requirements removed at beginning of the project:

During the *second sprint*, breakout 4 we learned more about requirement engineering. We had chance to done a thorough evaluation of our original requirements. We found *chatting functions* and *Game Publisher's Client* these two requirements are almost impossible to fullfill.

For the chat function, it is because the communication structure we choose for our two clients cannot carry instant messaging. If we insist on adding this function, we will have to write a lot of socket code just for this function. After we finished the final product we re-evaluate the feasibility of implementing this function, we came to a conclusion: If we had chosen to make both clients using same web framework, there are a large number of out of box APIs that we can use for this function. But because we chose to build a web page and a local client at the same time, the instant messaging function needs to be completely written by ourselves.

For the game publisher's client we decided to remove the requirements mainly because there's very little function in this client. We didn't thought about making a payment system nor a social platform for developers, the only function for the client would be to upload the game. We thought this was a distraction from our mission statement in the first place, so we removed the requirement.

Requirements that we did not finished:

Tagging the game, instant playtime display, realtime game play detail display are three requirements we did not accomplished at the end.

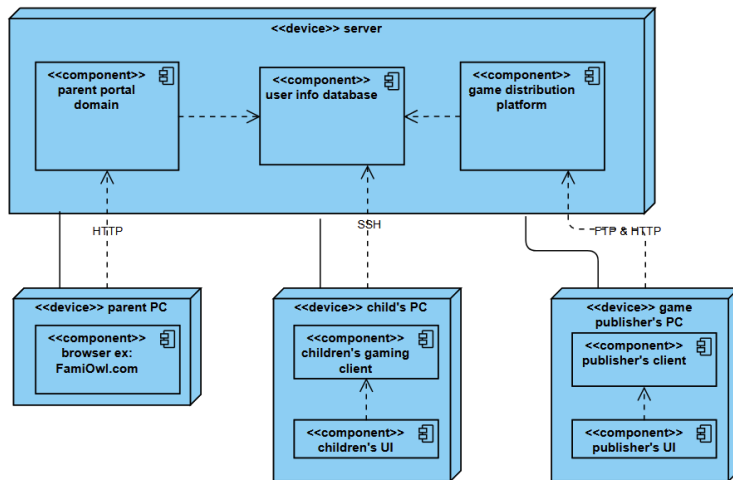
One of the features we haven't done is adding the game label function, and its estimated production time is around three days. Implementing this requirement includes: redesigning the game card ui, adding an input tag channel, Optimize label display order, checking for duplicate tags in the backend, and adding a search by tag function in the search bar.

Instant playtime display and displaying the game each child is currently playing are two functions we couldn't complete by the end of this sprint. They have similar reason as the chatting function. Because of the way we designed the server and the kid's client, most of the data transfer happens when the user signs off from the client, which is asynchronous query processing, and it passes the time they play the game, the time they add to the inventory and the things they play like. It is some what clumsy but it makes our client run smoother. Display the playing activities requires the instant transmission of data. We did not recognize this difficulty until we started to try to implement it.

Reflexion:

That's all the requirements that we didn't complete at the end of this project. Most of the reason we failed those is because we didn't realize that the asynchronous data processing technology we use prevents us from implementing the functions that require immediate transmission, and a small part is because we don't have better time management and team management to complete longer tasks

(communication between two clients)



- List the contributions of the software product and self-evaluation of the software product.
 - This software product is a convenient and powerful platform that allows users to browse a wide variety of video games and download their favorite video games easily and conveniently on their computers.
 - This product helps parents educate their children and prevent kids from addicting in games.
 - This software product provides convenience to a group that makes up a large percentage of society and meets the need for prevention of video game addiction. At the same time, the software is expandable (adding different kinds of games), easy to download (small overall memory consumption), and aesthetically pleasing, etc.
 - This product has strong reusability. In the future, our platform will become a diversified game platform when the game distribution features are improved, and at the same time, our website can add more features to meet additional needs.
- Lessons Learned
 - Evaluate the possibility of complete a function before we write it down to the requirement list
 - factors that may affect the completion of the function
 - our ability to implement the code
 - technical restrain (what is available to us)
 - Meet with objectives
 - we often scheduled meeting together just to code. this is not right, not the purpose of meeting

- low productivity & waste of time
- need to spend time together to discuss important issues
 - how we gonna navigate our project in the following week
 - problems we faced
 - how to solve the problem
 - goal for next week
- Give up miscellaneous details when there is more important to do
 - on the last few days of last sprint we still have some important issues regarding the shutdown function
 - even it's the last few days for us to actually code on our program we still put too much focus on perfecting the exact detail we planned out to that function, that is to shut down synchronously
 - it was unrealistic to complete synchronous shutdown
 - we should focus to deliver the functionality out first then to think how to perfect it