RETROWARE

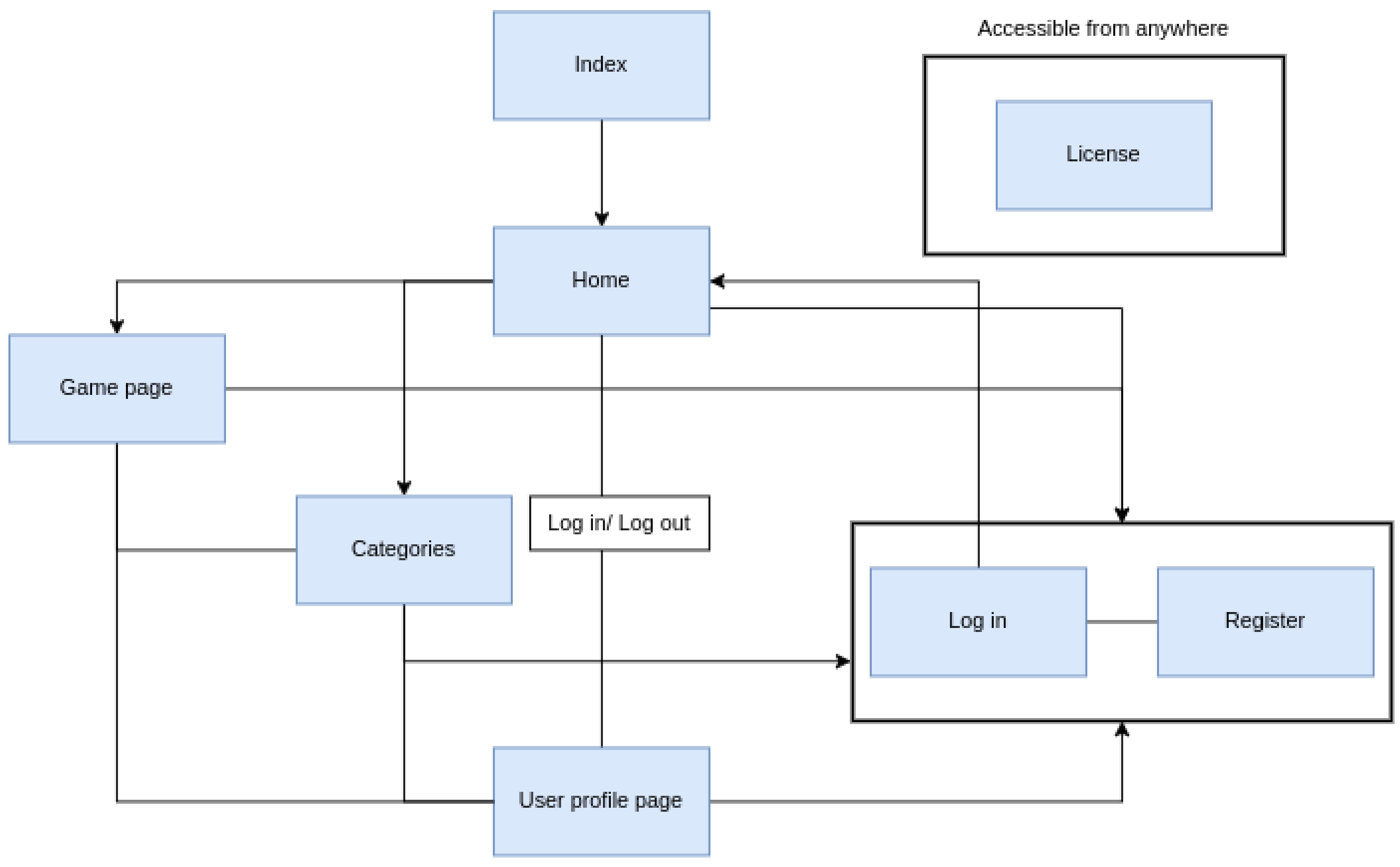
Alexandru C.

Table of contents

1. Application description

Retroware is an online in-borwser games website. Users who are also registered and logged in can leave comments, rate games and personalise their profile.

2. Site map



3. Usability and design

The website was implemented in small increments. Each small functionality was coded then tested in 2 browsers (Chrome and Firefox). At first static webpages were written, because it was quicker and less resource intensive, then these pages were translated into an XHTML source that JSF can interpret.

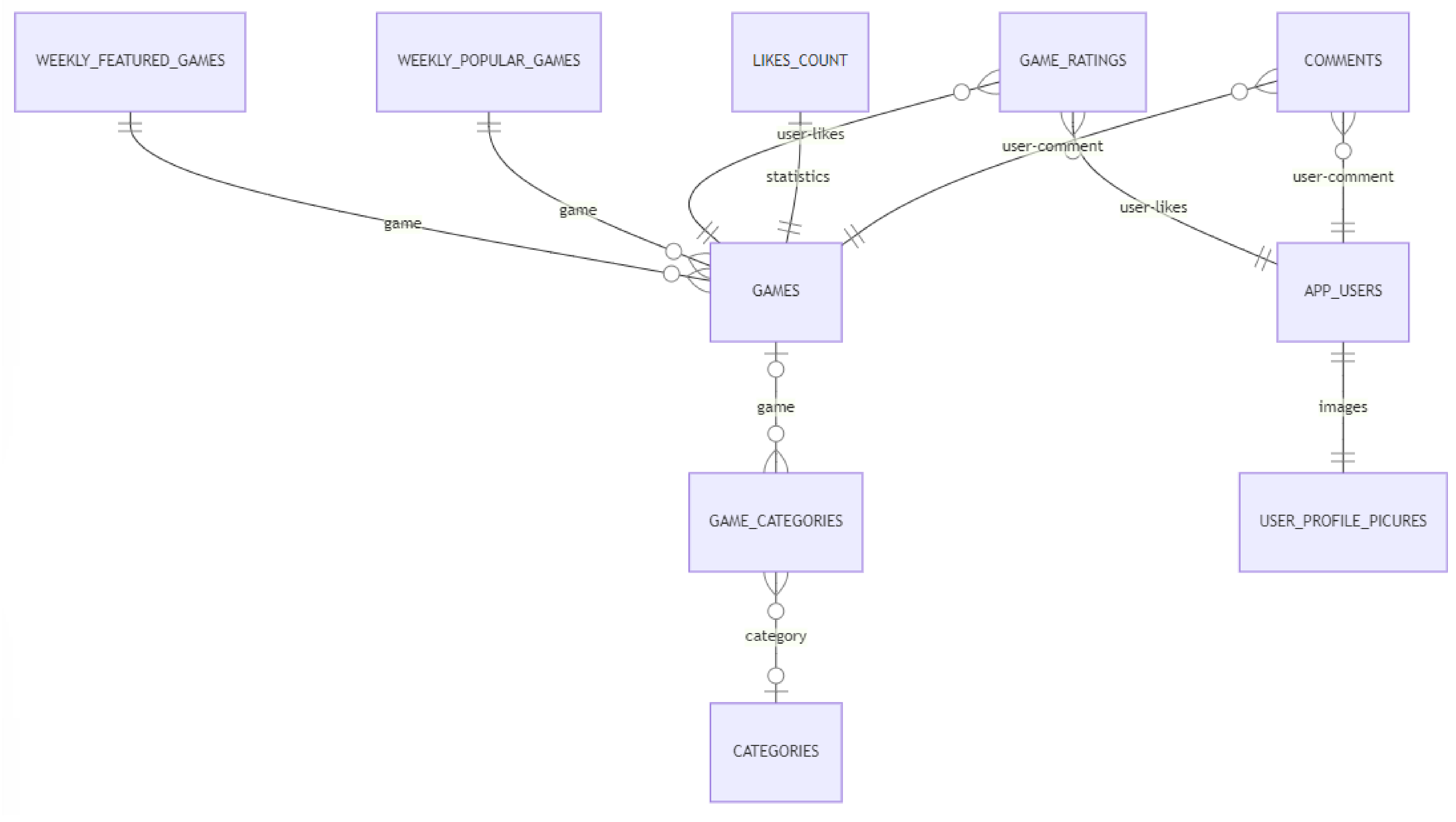
Each ManagedBean manages a specific area of the website and their scope is limited to just long enough to fulfil their purpose. For example the "GameView" bean is only concerned with game data, manipulating likes, dislikes, and comments while "UserAgent" is a "SesionScoped" bean which has to keep track of who the user is (what user).

Retro aesthetics were adopted as it reminds the users of 80s arcade machines, a fantastic time for small yet fun games. All resources were chosen carefully to build this aesthetic, including the animation on the index page which was animated using Blender 3D software.

Retroware's interface was designed so its intuitive and appealing. Webpages are also responsive meaning they can even be viewed on most mobile devices. The minimum recommended resolution is 800 pixels in width. Below this, the website will still display correctly however some of the games might not respond to input correctly.

4. The database

4.1. Entity relationship diagram



4.2. Database table structures

APP_USERS

Column name	Data type	Primary key	Foreign Key
USER_ID	INTEGER	YES	NO
USERNAME	VARCHAR(25)	NO	NO
EMAIL	VARCHAR(90)	NO	NO
PASSWORD_HASH	VARCHAR(32)	NO	NO
JOIN_DATE	DATE	NO	NO
DESCRIPTION	VARCHAR(150)	NO	NO
BANNED	CHAR(1)	NO	NO
SALT	BLOB(16)	NO	NO

CATEGORIES

Column name	Data type	Primary key	Foreign Key
CATEGORY_ID	INTEGER	YES	NO
TITLE	VARCHAR(25)	NO	NO

COMMENTS

Column name	Data type	Primary key	Foreign Key
GAME_ID	INTEGER	NO	NO
USER_ID	INTEGER	NO	NO

games

Column name	Data type	Primary key	Foreign Key
GAME_ID	INTEGER	YES	NO
TITLE	VARCHAR(25)	NO	NO
DESCRIPTION	VARCHAR(150)	NO	NO
PLAY_COUNT	BIGINT	NO	NO

Column name	Data type	Primary key	Foreign Key
RATING	SMALLINT	NO	NO
UNLISTED	CHAR(1)	NO	NO
RESOURCE_ID	VARCHAR(32)	NO	NO
PUBLISH_DATE	DATE	NO	NO

GAME_RATINGS

Column name	Data type	Primary key	Foreign Key
GAME_ID	INTEGER	YES	YES
USER_ID	INTEGER	YES	YES

LIKES_COUNT

Column name	Data type	Primary key	Foreign Key
GAME_ID	INTEGER	YES	YES
LIKES_COUNT	INTEGER	NO	NO
DISLIKES_COUNT	INTEGER	NO	NO

USER_PROFILE_PICTURES

Column name	Data type	Primary key	Foreign Key
USER_ID	INTEGER	YES	YES
IMAGE	BLOB(2097152)	NO	NO
IMAGE_FILE_NAME	VARCHAR(30)	NO	NO

WEEKLY_FEATURED_GAMES

Column name	Data type	Primary key	Foreign Key
GAME_ID	INTEGER	YES	YES

WEEKLY_POPULAR_GAMES

Column name	Data type	Primary key	Foreign Key
GAME_ID	INTEGER	YES	YES

5. Java classes explained

5.1. CategoriesView

This class is a view scoped managed bean. It is responsible for loading games into a list according to the categories that the games belong to, sorting them based on a few criteria such as highest or lowest rating. This bean also contains search methods that help the user find the a game they might be looking for.

5.2. GameView

The "GameView" class is a view scoped managed bean. It loads the content of a particular game (the one the user navigated to) and displays it. It also keeps track of like and dislikes, game play count and comments.

5.3 HomeView

A view scoped bean that loads 3 game lists: featured games, popular games, and all games. It does nothing more than hold these lists.

5.4. SearchRequest

A request scoped bean that holds a search string from a source pace (one that contains the navigation bar) and passes it to the categories page. The data it caries is released once the request is over.

5.5. UserAgent

This is a session scoped managed bean that keeps track of who the user's logged in state and what account is associated with the user (if the user is logged in). It manages logging in and logging out. It also keeps in memory what games the user liked to make the application more responsive.

5.6. UserPageView

A view scoped managed bean that loads user profile details and manages edits to the profile if the user loaded inside the "UserAgent" bean is the same as the loaded user.

5.7. UserRegisterForm

A view scoped managed bean that records form inputs, validates them and creates a new account if the the details are correct. This managed bean explicitly implements validator methods to deal with all the complications of validation in JSF.

5.8. BackgroundJobManager

An EJB bean that schedules jobs that the server carries out independently of the user. This class only has 1 scheduled job which is to update ratings.

5.9. Data container classes

These classes are the following:

- CategoryRecord
- CommentRecord
- GameRecord
- UserRecord

Their only purpose is to hold data fetched form the database for ease of use. They are implemented explicitly, not auto-generated.

5.10. SHA1PasswordHash

This is a utility object that is used to hash passwords before storing them in the database. Since hashes can not be reversed, it is also used to hash the login password before checking it against the stored password hash.

A method for generating salts is also present in this class. Salts are random byte values that help the hashing algorithm obfuscate the hashed password so they are much more difficult to crack.

5.11. UserProfileImageServlet

This is a servlet that is responsible for receiving GET and POST requests from the user page. It receives a user id and returns an image associated with the provided id. All images fetched originate from a database.

5.12. Validator classes

These classes are:

- EmailValidator
- PasswordValidator
- UsernameValidator

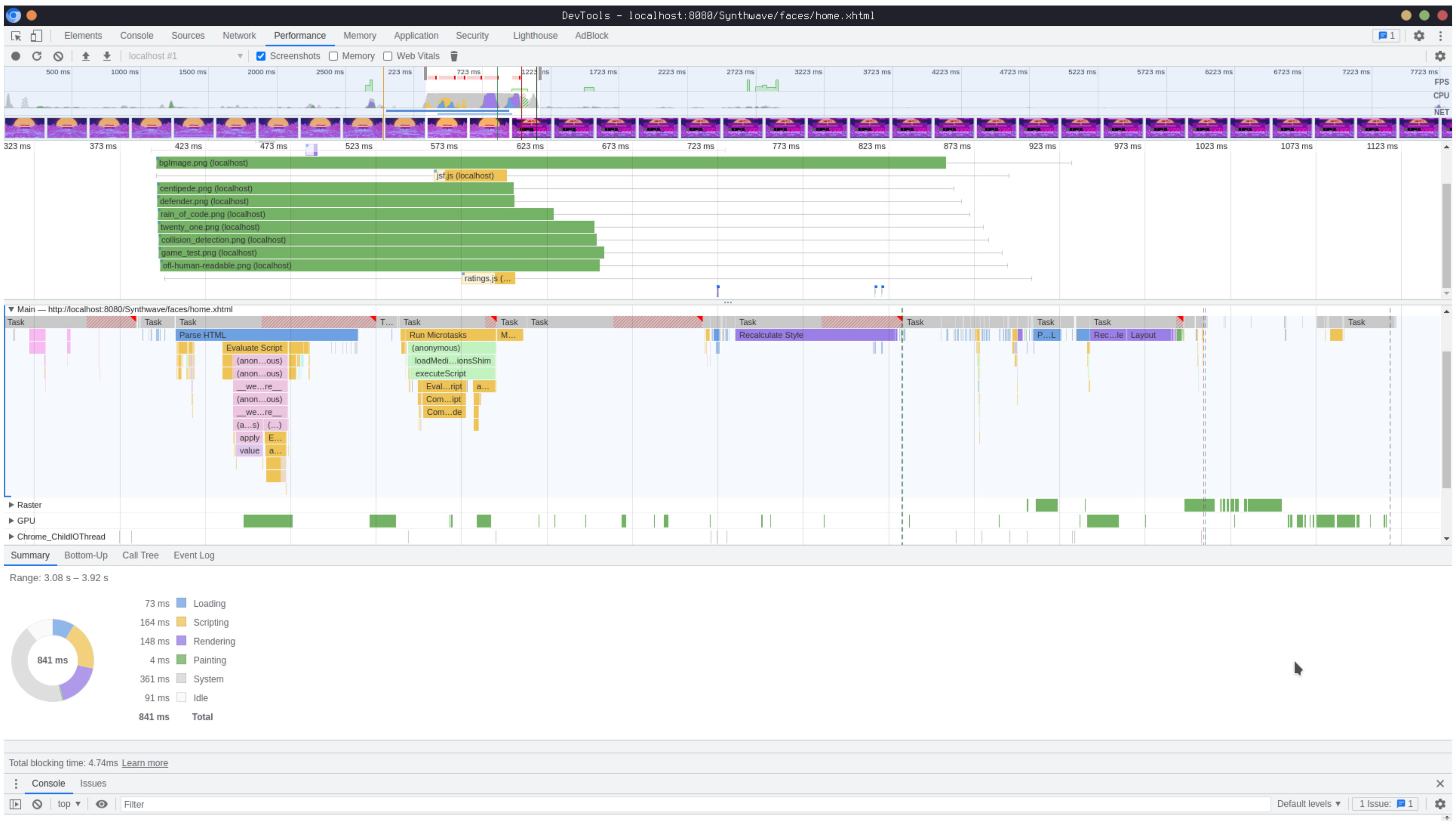
They can be used to validate fields while also offering more control over. It handles much more specifically validation and the messages it generates when an error occurs.

6. Testing

The website was originally developed on a Linux based machine and tested in 2 browsers. Everything worked as intended on the Linux machine however when running the application on Windows issues appeared. These issues are:

- background manager object will not update game ratings accordingly (fixed issue)
- user page will not automatically refresh when updating profile pictures

The performance of the website is important. One of the first issues with the web application was the long waiting time to get from the index page to the home page. This was caused by loading the background image of the website which was stored as a PNG (2.5 MB).

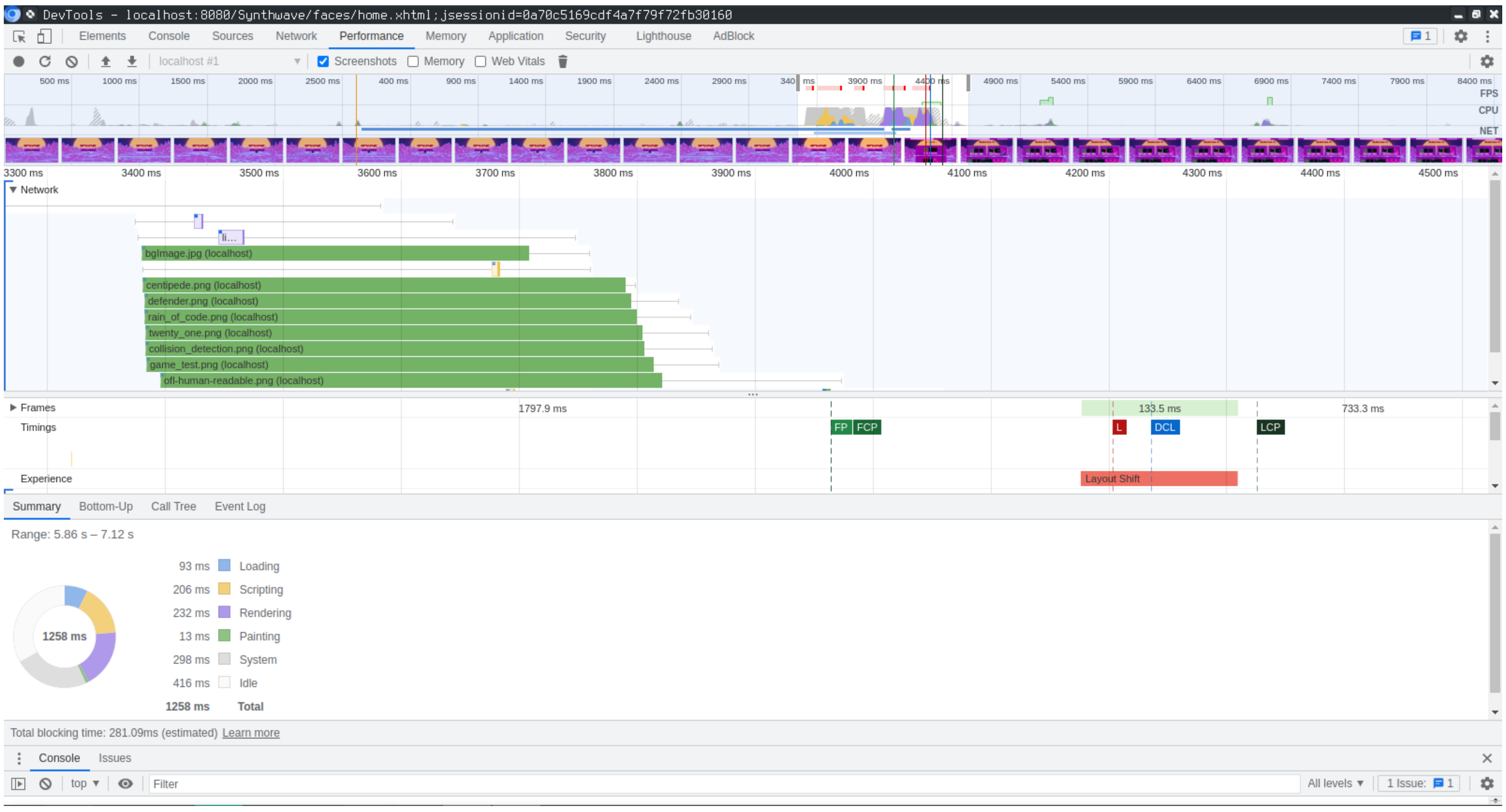

DevTools - localhost:8080/Synthwave/faces/home.xhtml

Range: 3.08 s – 3.92 s

73 ms	🟦 Loading
164 ms	🟨 Scripting
148 ms	🟪 Rendering
4 ms	🟩 Painting
361 ms	⬜ System
91 ms	⬜ Idle
841 ms	Total

Total blocking time: 4.74ms Learn more

This background was converted to JPG and it decreased significantly in size (274 KB).


DevTools - localhost:8080/Synthwave/faces/home.xhtml;jsessionid=0a70c5169cdf4a7f79f72fb30160

Range: 5.86 s – 7.12 s

93 ms	🟦 Loading
206 ms	🟨 Scripting
232 ms	🟪 Rendering
13 ms	🟩 Painting
298 ms	⬜ System
416 ms	⬜ Idle
1258 ms	Total

Total blocking time: 281.09ms (estimated) Learn more

The top most green bar in the previous images represent the background image loading time. In the second picture the loading time for the background image is noticeably smaller.
In fact converting all images that are not required to be high fidelity to jpg and scaling them appropriately will improve performance.