Habitat Connect

Laura Rodríguez López
Department of Computer Science
Hanyang University
Madrid, Spain
1postlaura@gmail.com

Christian Alexander Hernandez
Department of Computer Science
Hanyang University
Dallas, United States
ch407541@gmail.com

Marcel Plesky
Department of Information Systems
Hanyang University
Zürich, Switzerland
marcel.plesky68@gmail.com

***Abstract*—Habitat Connect is a web application designed for residents of Hanyang Habitat, our apartment complex, to stay connected, regardless of their cultural backgrounds. Our platform offers features like an announcement board, birthday calendar, washer checker, and maintenance request page. Residents can easily receive important updates, plan events, check laundry availability, and report maintenance concerns – all within a user-friendly interface. With Habitat Connect, we hope to foster a stronger sense of community and make everyday tasks more convenient for everyone in Hanyang Habitat.**

ROLE ASSIGNMENTS

In the table below, we can find a detailed breakdown of the roles and responsibilities assigned to each team member throughout the duration of the project.

Role	Name	Task Description
User/Customer	Laura	- Uses the software - Establishes the conditions and features that the software must satisfy (requirements) - Provides feedback - Approves change requests
Software Developer	Christian	- Works with user/customer to gather their needs - Designs and develops the software - Tests and debugs the software
Development Manager	Marcel	- Supervises the software development process - Facilitates communication between user/customer and software developer - Manages tools and resources - Monitors progress and quality of the software

TABLE I: Roles and Responsibilities

I. INTRODUCTION

A. Motivation

When international students arrive at Hanyang University, they often seek affordable and short-term housing options. One of the most popular accommodations are Goshiwons like Hanyang Habitat. As members of this team currently living in Habitat, we would like to provide residents with a new toll that caters to their needs.

Living in Hanyang Habitat has given us firsthand insight into the challenges faced by exchange students. With approximately 70 international residents that come from more than 15 countries, it's evident that online communication platforms vary widely. For instance, Europeans prefer applications like WhatsApp or Instagram, while North Americans rely heavily on text messaging. In contrast, WeChat is very popular in certain Asian countries like China or Indonesia. In South Korea, KakaoTalk is the go-to messaging app, but its unique features and interface can pose a learning curve for newcomers, especially for those that do not speak Korean.

B. Problem Statement

Given these cultural differences regarding communication tools, we asked ourselves: which platform should Hanyang Habitat residents use to effectively communicate with one another? Our proposed solution is Habitat Connect, an announcement board with a user-friendly English interface.

As well as serving as an online communication tool, Habitat Connect offers more features that focus on the needs of Habitat residents. This includes a laundry checker that provides real-time updates on washer availability, a birthday calendar to keep everyone informed about upcoming celebrations, and an easy-to-use maintenance request system, ensuring a direct line of communication between the tenants and the apartment manager.

Habitat Connect ensures that all residents, regardless of their language and online backgrounds, can comfortably navigate and utilize the tool. This idea comes from a desire to foster a sense of community among students in Hanyang Habitat. We aim to enhance the living experience of international students, making their time at Hanyang University more comfortable and rewarding.

C. Research on related software

WHATSAPP

WhatsApp is a widely used messaging application that offers instant messaging, voice and video calls, file-sharing and even a poll-system that allows voting for user given options. WhatsApp is used in family, friends and even work environments to stay connected and up to date with your contacts.

KAKAOTALK

Just like WhatsApp, KakaoTalk is a popular messaging application. It offers even more features than WhatsApp like mobile payment and other various app services making it a multifunctional communication platform.

DISCORD

Discord is known for its features like text and voice chat, creating and managing channels, and building communities around specific interests or topics. It's often used for both casual and structured discussions, making it a versatile option for promoting communication and collaboration in a variety of settings, including apartment buildings and shared living spaces.

GOOGLE CALENDAR

Google Calendar is a user-friendly online calendar application from Google. It allows users to create, manage and share events, streamlining planning. Moreover, users can also import birthdays from their contacts, making it easier to remember important dates. Google integration and various viewing options make it a valuable time management and organization tool.

FAMILYWALL

FamilyWall is an application designed to help families and individuals organize their lives, share information, and manage family-related activities. It allows users to create shared calendars, to-do lists, shopping lists, and share photos, as well as important information with family members or within their respective groups.

LAUNDRY APPLICATIONS

Some laundromats like Samsung and LG have developed their own mobile apps to check the availability of washing machines and dryers.

ZENDESK

Zendesk is a popular ticket management system used for customer support and issue tracking. It helps organize and track customer inquiries and support tickets. With Zendesk, businesses can provide better customer service.

II. REQUIREMENT ANALYSIS

A. Register via Local Registration

Upon entering Habitat Connect, users will encounter a "Register" button situated in the top right corner of the screen. Clicking on this button redirects the user to the registration page, where Habitat Connect offers support for both local and Google OAuth registration. If the user chooses to register via local registration, they will be asked for a valid email address and a password. Once the user initiates the registration process, the inputs are processed internally. That is, we perform the checking of a valid email address and password format. If the validation is successful, we create the user's account within our database.

The next step of the registration process redirects users to the "Complete Profile" page, where they need to provide additional details to set the profile account. Users will provide a username, birthdate, and Habitat's room number. The registration process is complete when the user clicks on the "Complete profile" button, successfully passing the internal input validation, and is then redirected to the main announcement page of Habitat Connect.

B. Log in via Local Registration

Upon entering Habitat Connect, users will notice a "Log in" button situated in the top right corner of the screen, positioned to the left of the registration button. Clicking on this button redirects the user to the login page, where Habitat Connects offers support for both local and Google OAuth login processes. If the user chooses to login locally, they will be asked for an email address and password. Then, the system conducts an internal search for the user in the database. If the user is found, they will be seamlessly redirected to the main announcement page of Habitat Connect. On the other hand, if the user credentials are not registered in the system's database, the user will be prompted to provide alternative credentials

C. Register via Google OAuth

When users choose to register with Google, they engage in Google OAuth, a secure process simplifying registration. This method lets users authorize third-party apps like Habitat Connect, using Google credentials without revealing their password. Users initiate this process by clicking "Register with Google" on Habitat Connect's registration page. After selecting their Google account, they may need to log in if not already. Google prioritizes user privacy, presenting requested permissions for approval. Once granted, Google generates an OAuth token, confirming identity and consent. Our system creates the user's account in our database without further validation, leveraging Google's checks. It is important to note that when users register via Google OAuth, their Habitat Connect account lacks a password field. As a result, users who register with Google OAuth can only login using Google.

The next step of the registration process redirects users to the "Complete Profile" page, where they complete their profile setup. From here, the process is the same as for the local registration.

D. Log in via Google OAuth

Logging into Habitat Connect with Google OAuth offers users a host of advantages, with convenience, security, and efficient access management being chief among them. This approach eliminates the cumbersome need to remember and juggle numerous usernames and passwords for different online platforms, simplifying the user experience significantly. Furthermore, users maintain a high level of control over their digital footprint, with the ability to easily revoke access to specific services through their Google account settings whenever they deem it necessary. This synergy between ease of use and robust data security underscores the appeal and practicality of logging in via Google OAuth. Once Google successfully completes the login validation, users are seamlessly redirected to the main announcement page of Habitat Connect.

E. Dashboard

The dashboard is the main page. The user is able to see all announcements ever posted, except those deleted by their corresponding authors. Regarding announcements, the following actions are available for the user:

- **Post** an announcement: users can write announcements that can only contain ASCII characters. Once they choose to post the announcement, it will be available in the dashboard to every user registered in Habitat Connect. The ownership of the announcement is registered as well..
- **Edit** an announcement: users can edit the title and/or body of their posted announcements.
- **Delete** an announcement: users are able to delete their posted announcements. They will no longer appear in the dashboard, and they will be removed from the system's database.
- **Comment** on an announcement: registered users can comment on posted announcements. Comments can only contain ASCII characters. Furthermore, the ownership of the comment is available for every user to see.
- **Edit** a comment: users can edit their posted comments.
- **Delete** a comment: users are able to delete their posted comments. They will no longer appear as comments in the announcement view, and they will be removed from the system's database.
- **Search** an announcement: users can search for announcements by entering keywords, which are then searched in both the title and body of the announcement.

F. Calendar view

The system will provide a calendar view for users that automatically displays other users' birthdays based on their sign-up information. The calendar view will be monthly, and it can be switched between months with the help of navigation arrows on each side of the calendar. Users should be able to click on a specific date on the calendar to view the list of birthdays for that date, including the names of the users celebrating their birthdays.

G. Laundry checker

This function will be on the left side of the main page in the form of an icon, and once clicked, users will be taken to another page where they will see each washer, which is assigned a unique number. The status of each washer is displayed in the Status column, with different colors to make it visually clear: IN USE is displayed in red, AVAILABLE is displayed in green, and FINISHED is displayed in white. Visually, it will be in a table-like format for easy understanding. Each washer has a set of action buttons next to it. The available actions are:

- AVAILABLE (green button): users can click this button when they are going to use the washer. Once clicked, the washer's status will change to IN USE, in the color red.
- FINISHED (white button): once a user is done using the washer, they will click a button that will change the status back to AVAILABLE, displayed in green.

There may be additional features such as timers to display how long a washer has been in use, as well as user-friendly tooltips or information icons that can provide additional information or instructions.

H. Maintenance requests

Under the laundry checker icon, we can find the Maintenance Request icon. Once it is clicked, users will be taken to a new page. There will be two text boxes to create a Ticket: the first textbox will be the ticket name (for example "No Hot Water"), and the second text box will be the description of the maintenance issue with a maximum of 200 characters. Once the ticket is set up, there will be a Submit Ticket button at the bottom of the page. Once clicked, an email will automatically be sent to the apartment manager's private email. The page for the user after clicking the Submit Ticket button will also change. In the first half of the page, we will find a message saying "Your ticket has been submitted!". Under that will be the status of the ticket which can be one of two options: Under Review (which will automatically be selected), and Ticket Completed. Once the user's request has been completed in person, they can click the Ticket Completed option on the Maintenance Request page, which will update the user's page back to the original create a ticket page.

I. User feedback Support

In order to manage the website without bugs, we need a substantial amount of feedback from users. This is a function that most every high level website has in order to enhance user experience. At the top right of the screen, where "Profile" is, one of the options under above "Settings" is a "Report Bug" option. Once clicked, the user will be taken to a separate page, where similar to an email, will have a Title and description text box with a maximum of 300 characters. Once both are filled and the user clicks "Submit", an email will automatically be sent to an admin regarding the bug. This will also be used as a "Support Ticket", whenever someone needs to get in contact with an admin regarding an issue they have about their account.

J. User profile options

The user profile is a fundamental component of an application, serving as a repository of user-specific information and a platform for users to manage and personalize their experience within the system. The user profile typically encompasses several key options and functionalities designed to cater to the diverse needs and preferences of the application's users. However, to keep the concept of this web application, the changes the user can make to their profile is limited. On the "Profile" section and "Edit" subsection, the user will be taken to a new page, where they can apply one of 3 options regarding their account. The first option is to switch to anonymous mode, where their room number and name will be hidden. With this, the user is still able to read the announcement chat and use the washer checker, however they cannot type in the announcement chat or create a maintenance manager

ticket. The second option is to edit their username, as upon signing up for our website, their real names are used for the users accounts. The users only have the option to change their usernames once a month. The last option they have is to change their profile pictures. There is no limit to this option and the users can change it as many times as they please.

III. Development environment

A. Choice of software development platform

Given that Habitat Connect is a web application, the choice of operating system is quite flexible. That is, we can choose any operating system that supports web development tools. To be more specific, our working environment consists of the following:

Name	Version	Description
Windows	11 Home	- User-friendly environment with an updated UI - Wide range of web development tools and IDEs
macOS	12.5 Apple M1 Pro	- Unix-based - Offers a powerful terminal for web development

TABLE II: Working environment

Moreover, our development process will make use of some tools that can be found below.

1) Node.js is a server-side runtime environment for executing JavaScript code, enabling efficient web application development. It excels in handling real-time interactions and concurrent requests

Fig. 1: Node.js logo

2) Bootstrap is a front-end framework that provides predesigned, responsive, and customizable UI components. It allows us to create appealing and web applications with ease.

Fig. 2: Bootstrap logo

3) MongoDB is a NoSQL database that provides a flexible, document-based data structure. It's designed for handling large amounts of unstructured data, making it suitable for dynamic web applications.

Fig. 3: MongoDB logo

4) Visual Studio Code is a powerful code editor that supports various programming languages. Furthermore, this IDE offers features like syntax highlighting, debugging, extensions, and Git integration.

Fig. 4: Visual Studio Code logo

5) Git/GitHub: Git is a version control system that allows multiple developers to collaborate on a project. On the other hand, GitHub is a web-based platform that provides Git repositories, collaboration tools, and hosting services. It facilitates efficient team collaboration and code management.

(a) Git logo

(b) GitHub logo

Fig. 5: Logos for Git and GitHub

6) Google Drive is a cloud-based storage and collaboration platform. It allows us to store files, share documents, and collaborate in real-time. It's useful for storing mock documents that will be pushed to GitHub.

Fig. 6: Google Drive logo

7) Overleaf is an online platform for collaborative document editing. It simplifies the process of creating and editing scientific and technical documents, so it's very popular among researchers and academics. We will make use of it to create our LaTeX documents.

Fig. 7: Overleaf logo

8) MOQUPS is a web-based design and prototyping tool that allows users to collaborate on UI design in real-time

Fig. 8: Moqups logo

Regarding the choice of programming language, we will be using the following:

1) JAVASCRIPT is a high-level programming language primarily known for its use in web development. It allows us to add interactivity and dynamic behavior to web pages. With Node.js, it can also be used for server-side development.

Fig. 9: JavaScript logo

2) HTML is the standard language used for creating web pages. It provides the structure and content of a webpage, using elements like headings, paragraphs, and links.

Fig. 10: HTML logo

3) CSS is a language used for styling web pages. It controls the visual presentation of HTML elements, including aspects like layout, colors, fonts, and animations.

Fig. 11: CSS logo

All software development tools are free of use. Furthermore, since Windows11 and macOS are already in use, we will not count them as costs.

B. Software in use

Today people rely on widely-used software like Discord for chatting, Google Calendar for planning, and FamilyWall for staying organized. Our upcoming project takes inspiration from these popular tools and blends them into one. Our software will make it easy for people living in the same place to chat, schedule events, and share information, enhancing the way they interact and stay connected.

C. Task distribution

In such a collaborative project among three students, roles have been thoughtfully distributed to capitalize on our individual strengths. We believe that not dividing the tasks will result in an inefficient and not realistic way of software development. The workload can be divided into distinct categories, each focusing on a crucial aspect of Habitat Connect. The tasks are as follows:

- **Management**: responsible for checking up on each team member work, ensuring adherence to deadlines, and coordinating the overall workflow
- **Documentation**: responsible for writing the pdf and creating the LaTex files, ensuring to present a well-structured and comprehensive overview of the project
- **Frontend**: charged with the responsibility of developing the frontend side of the software. This includes crafting the user interface, ensuring a seamless and interactive user experience, and design and implement every component for user engagement
- **Backend**: charged with the responsibility of developing the backend side of the software. This includes handling the database, server-side logic, and ensuring that every software component correctly adheres to all specified requirements

After thoughtful evaluation, the final assignment of tasks to each team member has been documented in the following table:

Name	Role	Responsibilities
Laura	Backend Frontend Management Documentation	As the backend developer, this member is responsible for constructing the software foundation code. This involves turning our project requirements into code, integrating external libraries to enhance our project capabilities, and addressing bugs to maintain a robust and functional backend infrastructure. As frontend developer, this member undertook the responsibility of designing and crafting a basic frontend infrastructure, providing a solid starting point for the rest of the team. Moreover, this member's management tasks included taking responsibility for overseeing the team dynamics and workflow. Not only involves a checking up on each team member's work but also ensuring strict adherence to deadlines. Additionally, as a documentation assistant, this member was charged with the task of reviewing documentation for clarity, suggesting ideas for improvement and organization, and actively participating in the creation of LaTex files.
Marcel	Frontend Documentation	As a frontend developer, this member assumed the responsibility of reshaping the user interaction within the software system. This involves not only the rearrangement of Bootstrap elements but also the incorporation of fresh and engaging content that contributed to an enhanced and visually appealing user experience. This member approach was not merely aesthetic but strategic, aiming to create an immersive and visually appealing environment that resonated with end users. Moreover, this member's tasks also included an active participation in the documentation tasks, providing critical assistance to the main documentation writer.
Christian	Frontend Documentation	In his role as a frontend developer, this member's tasks included refining the user interface by further rearranging bootstrap elements. As the main documentation writer, he was entrusted with documentation responsibilities, ensuring that all elements are constantly updated and aligned with code changes. This approach to documentation ensures that the project's report remains accurate and reflective of the software's evolving codebase.

TABLE III: Task Distribution

IV. Specifications

In the section below, we will provide a detailed description of the implementation process of each requirement.

A. Sign up/Log in

If logged out, or a user is entering the website for the first time, the first component that users will see is the login page, as the website is a private application not open for public use. This page will have two main functions to enter the main webpage, sign up and login. The user sign up process on Habitat Connect is the starting point for users to establish their digital presence. It allows them to create unique usernames, set up secure passwords, provide email addresses, and specify room numbers. These steps collectively enable users to register and create a personalized digital identity within the platform, enhancing their overall user experience.

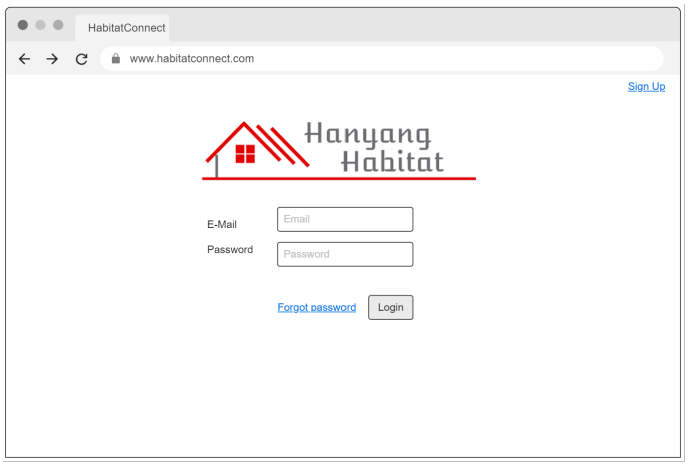

Fig. 12: Log in page

Log In

The user login process in Habitat Connect is the secure gateway to the platform, offering privacy and security for the users.

Correct login

If the correct information is inputted, the page will load to our personalized loading screen into the main Announcement Board page.

Incorrect password

Users will have a maximum of 5 password attempts. If exceeded, the user will be locked out for an hour. When an invalid password is entered, a message in red will popup. If the user forgets their password, they can click the "Forgot password" button where they will receive email notification associated with the account which will redirect the users to a page to create a new password.

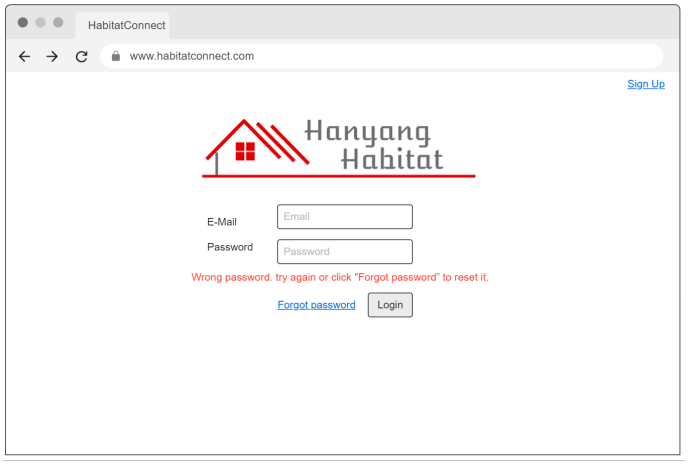

Fig. 13: Incorrect password popup

SIGN UP

Once the sign up button at the top right of the page is clicked, the user will be taken to a new page to fill out the required fields in order to login. To ensure a robust and personalized user experience on Habitat Connect, certain fields stand as mandatory during the sign-up process. These include room number, a valid email address, a password meeting specific security requirements, a unique username, and the user's date of birth. These essential details enable users to establish their digital identity, ensure secure access, and facilitate personalized interactions within the platform. Below, we can find the fields specification details:

- `Username`: maximum string of 12 characters. Special characters are allowed
- `Password`: must exceed 8 characters. Special characters allowed and must included 1 capitalized characters
- `Email`: must provide valid email
- `Birth date`: MM/DD/YYYY order, automatically inducted to the birthday calendar once signed up
- `Room number`: maximum 4 integers corresponding to tenants room number. No characters allowed

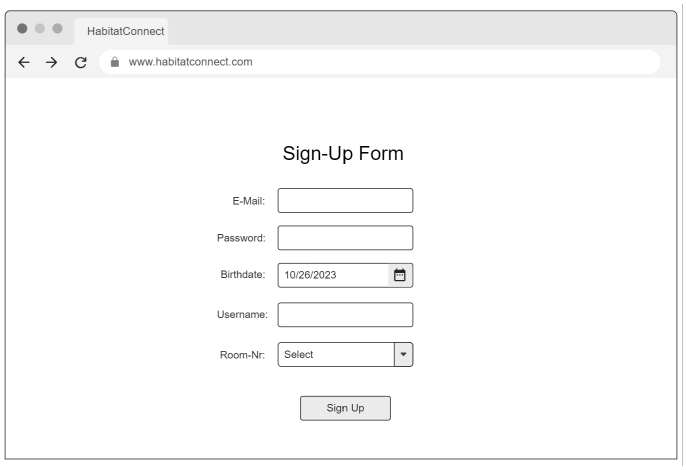

Fig. 14: Sign up form

B. Recent announcements

The announcement board will be the first thing that users will see upon logging into our web application. This will be considered our "Main Page". This screen will allow users to perform a specific number of actions.

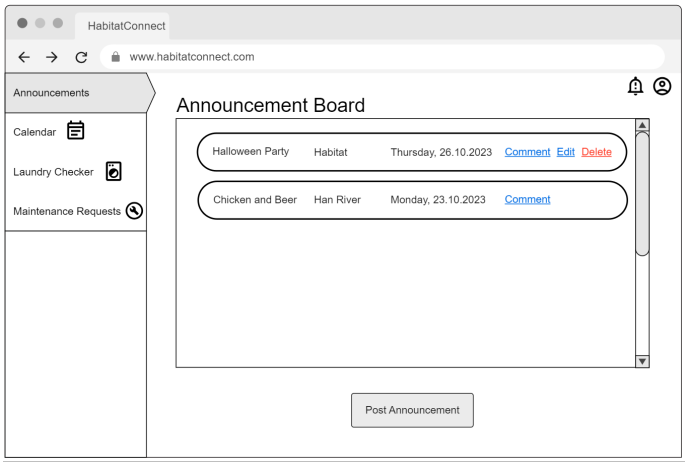

Fig. 15: Recent announcements page

Habitat Connect offers a range of communication and interaction features for its users. These features include the ability to post announcements, which can include text, as well as making information accessible to all registered users and ensuring transparency through clear ownership attribution. Users can also edit their announcements to keep them up-to-date and relevant, and they have the option to delete announcements if they become outdated or if the announcement was simply canceled. Additionally, users can engage in discussions by commenting on announcements, with comment ownership clearly visible, promoting accountability and a sense of community within the platform.

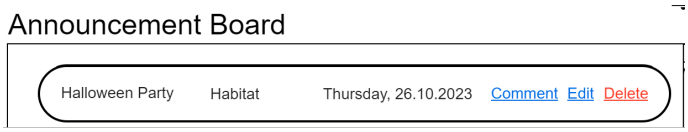

Fig. 16: Announcement options

POST ANNOUNCEMENT BUTTON

Once clicked, the user will be taken to a different screen, where they have to input 3 options: title, date, and location. All of these fields must be completed in order to post an announcement.

- `Title`: character string of maximum 50 characters
- `Date`: DD/MM/YYYY format
- `Location`: character string of maximum 70 characters

Once the user clicks the "Post Announcement" button, the screen will reload into the main Announcement page with the new announcement updated.

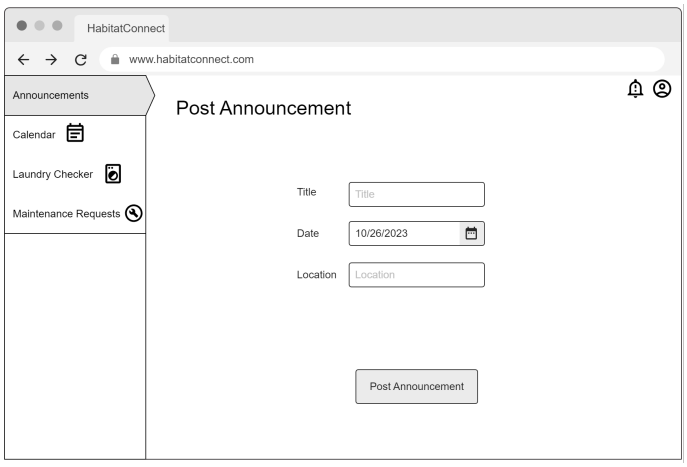

Fig. 17: Post announcement page

Edit Announcement button

This button will be visible directly on the announcement the user wants to edit. Once clicked, the user will be taken to another page, similar to the Post Announcement page where they can edit the title, date, or location. The only difference between this page and the Post Announcement page is the "Post Announcement" button being changed to "Save". Once clicked, the updated content will be visible on Announcement.

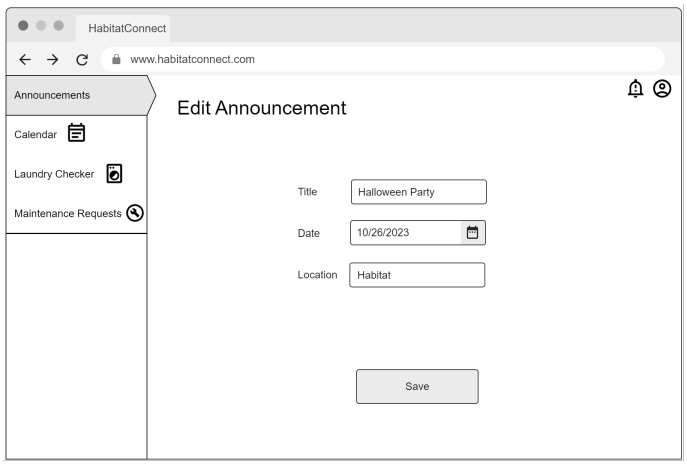

Fig. 18: Edit announcement page

Delete Announcement button

This button will be on the same row as the Edit Announcement Button. Once clicked, the user will be given a confirmation popup saying "Are you sure you want to delete this announcement?", with a Cancel/Delete option. If delete is clicked, the announcement will be deleted immediately without the page reloading, and the announcement below it will move up one.

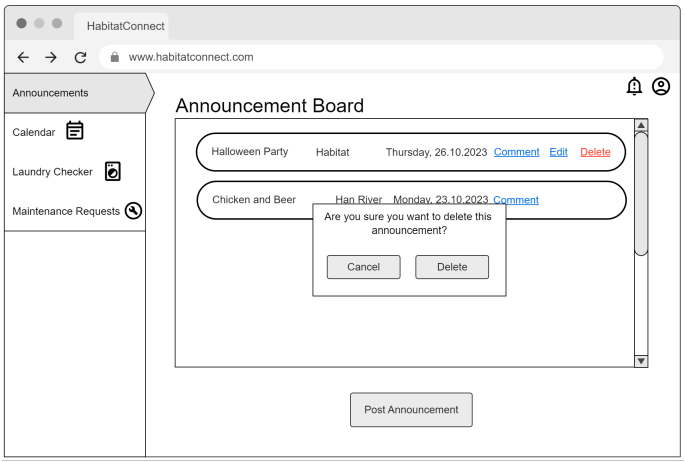

Fig. 19: Delete announcement confirmation popup

Comment button

This button will also be on the same row as Edit and Delete. Once clicked, the user will be taken to another page. In this page, the comment history of past users can be seen on the announcement, as well as a text box where users can type and post a comment, which will also be the only action available for the user in this page. The order will be from the latest comment to last.

- `Post comment text box`: character string maximum 150 characters

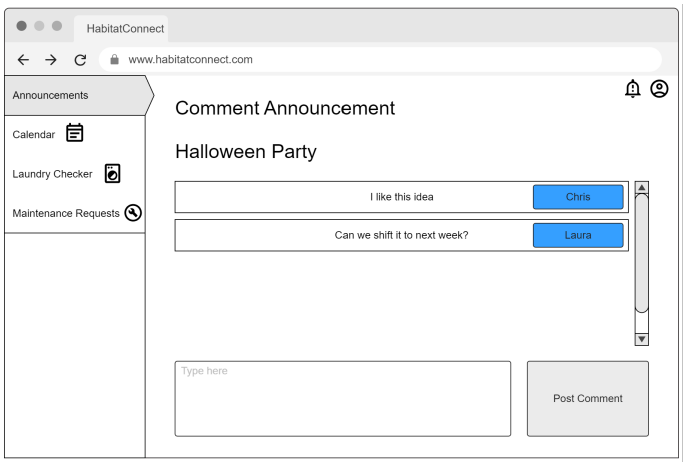

Fig. 20: Comment example

Once the "Post comment" button is clicked, the comment will be at the top of the list of comments.

C. Calendar view

The calendar view can be directly accessed through the task bar on the left side of the screen. It is a dedicated feature within the Habitat Connect platform, designed to commemorate and celebrate the special occasions of its users. It enables users to keep track of birthdays, create memorable moments, and foster a sense of community through shared celebrations. This page will not have any actions, and is more of a visual aid for

birthdays. Users will see the birthdays of the current month, but it is also possible to change the calendar view to display past/next months birthdays.

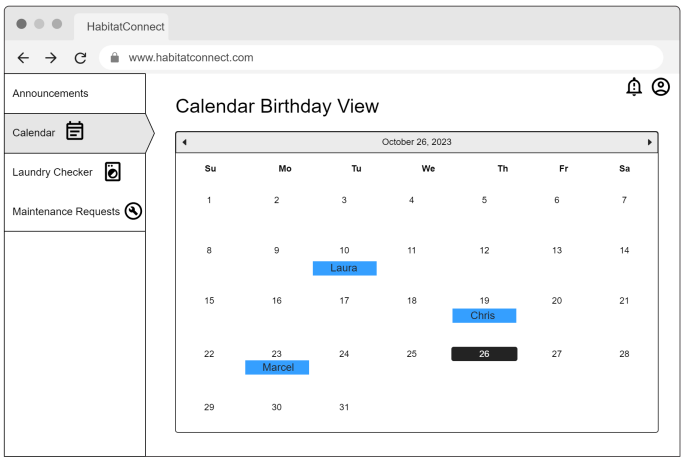

Fig. 21: Calendar view page

Upon clicking a birthday on the calendar, a popup will showcase, which allows users to see the detailed view of the birthdays.

Fig. 22: Birthday details

D. Laundry checker

Another one of the main four functions of the application is the laundry checker, which will have several actions that the user is allowed to do. Visually, it will consist of a table with three rows: washing machine number, current status, and change status. The change status row is the only row where users can perform an action.

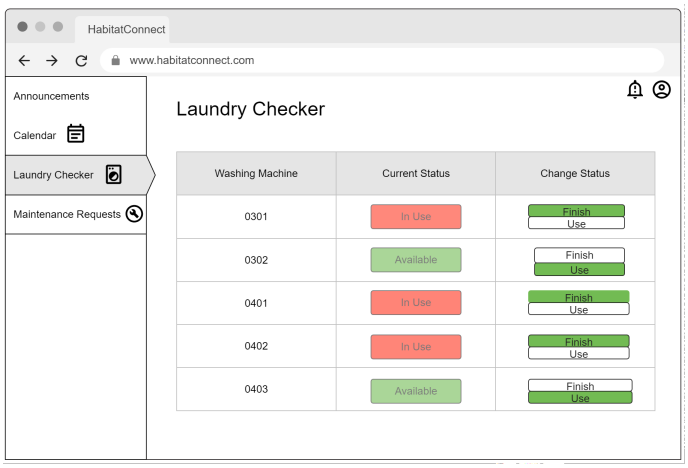

Fig. 23: Laundry checker page

Current status

There is no action to be done in this row. It is simply to show the status of the washer and is completely dependent on the "Change status" row. Moreover, if a user forgets to click the "Finish" button once their wash is finished, the status will automatically change to "Available" after 2 hours.

Change status

The users can perform two actions, either click the "Finish" button or the "Use" button. If the status is "In Use", users only have the option to click the "Finish" button. On the other hand, if the status is "Available", the user will only have the option to click the "Use" button.

- `Finish button`: only available when Status is "In use". Once clicked, the status will automatically change to "Available".
- `Use button`: only available when status is "Available". Once clicked, the status will automatically change to "In use".

E. Maintenance requests

Habitat Connect simplifies the process of submitting maintenance requests, streamlining the user experience. Users can conveniently log requests for repairs, improvements, or assistance, ensuring that their concerns are addressed promptly. The platform not only fosters efficiency but also enhances transparency by allowing users to track the status and progress of their requests. Upon opening the maintenance requests tab on the taskbar, the user will be taken to a new page in which they can perform a maintenance request. The requests made by the users contain the following fields:

- `Title`: string of maximum 20 characters
- `Description`: string of maximum 300 characters

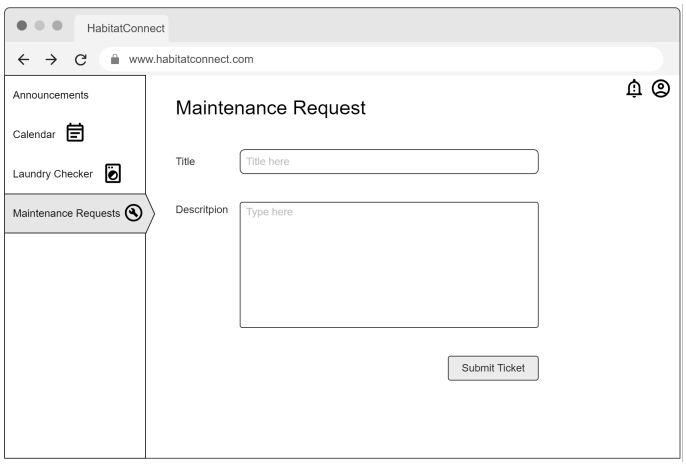

Fig. 24: Submit ticket page

Once the user clicks the "Submit Ticket" button, the page will update, confirming the user's maintenance request in the form of a ticket. A table will also be shown, showcasing the ticket ID, ticket status, and ticket title. No action can be performed once loaded to this page. Only one ticket is allowed at a time for a user. Furthermore, the ticket will automatically clear after three days.

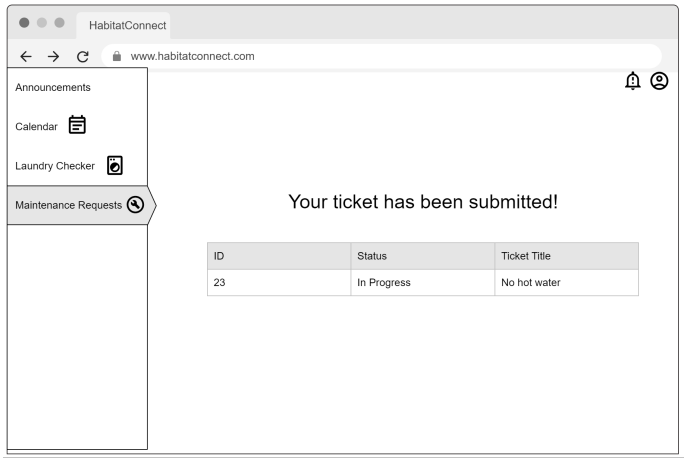

Fig. 25: Example of a submitted ticket in progress

The ticket, once submitted, will be automatically sent to an admin/apartment manager in the form of an email. At this point, the maintenance issue should be verbally deliberated between the tenant and admin. Once the maintenance request has been solved, it will be displayed as a completed ticket.

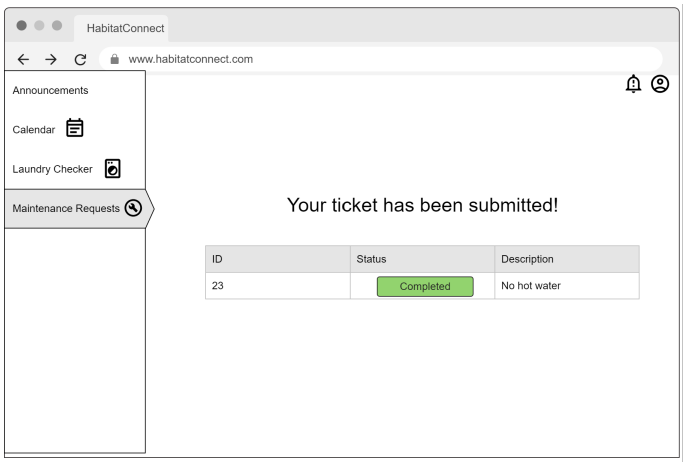

Fig. 26: Example of a submitted ticket solved

F. Loading page

Upon the initial launch of the page, users can expect a brief loading time of approximately 3 seconds before reaching the main announcement board. This short delay allows the various elements and objects on the web page to load smoothly, ensuring a seamless and enjoyable browsing experience. The loading screen will consist of our logo and a loading icon upon every reload. Scrolling between taskbar functions will not cause the main reload visuals to pop up.

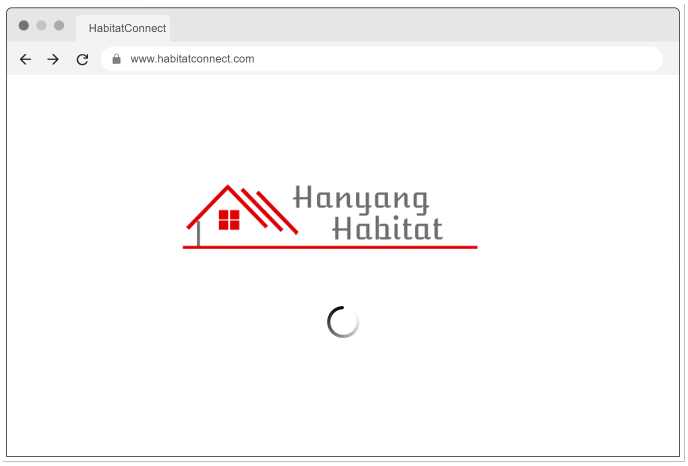

Fig. 27: Loading page

G. Notifications

Notifications are an essential part of a community platform to make sure all users are caught up on announcements and their maintenance requests, as well as each other's birthdays. For users to check their notifications, they must look at the top right of their screen and locate a bell with an exclamation point. Once they click that bell, they will see a listing of their notifications. In order to keep a simplistic approach and not overwhelm users with unnecessary notifications, they will only receive alerts from new announcements, the status of maintenance requests, and birthdays. Moreover, one of the

main causes for concern is if a announcement notification text is too long to fit in the bell's textbox, however, to compensate, if a notification strings surpass the max notification string of more than 13 characters, then on that 13th character, there will be a "..." to signify there is a longer message being transcribed.

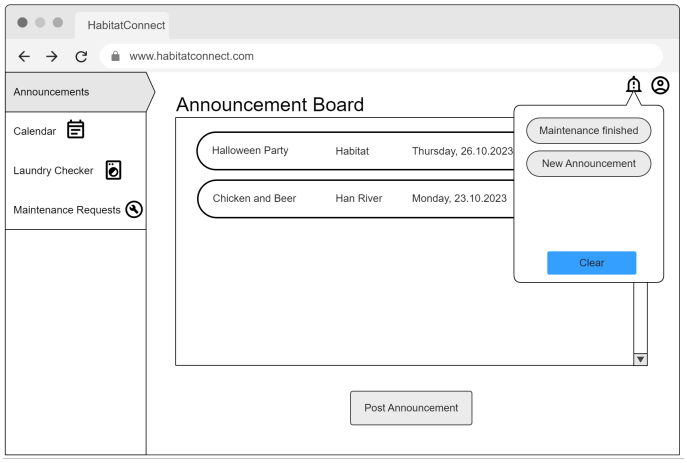

Fig. 28: Example of notifications

CLEAR BUTTON

To clear every notification in the bell box, users can safely click the "Clear" button at the bottom middle of the bell box. Once clicked, the notification will be completely cleared and cannot be retrieved again.

EXIT BUTTON

There is no actual exit button. Instead, to exit the notification bell box, users can click anywhere on the screen and it will dissipate the text box.

H. User feedback support

Customer feedback is highly valued and plays a pivotal role in our commitment to delivering an exceptional user experience. Input from our valued users is incredibly important, as it enables us to continually enhance services. If users have suggestions for new features, come across any issues, or wish to share their thoughts, they should refer to the "User Feedback Support" button at the bottom left of the web page. Developers will be weary of every feedback, as the submit button will make a new entry of bugs or possible new features for the admins in the database collection.

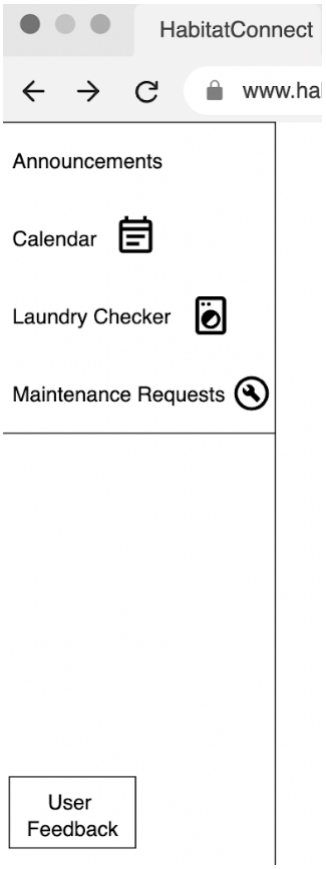

Fig. 29: User feedback button

USER FEEDBACK BUTTON

Upon clicking the "User Feedback" button, a text box will automatically appear. In this text box, a user can provide feedback or report bugs on the web page within 150 characters.

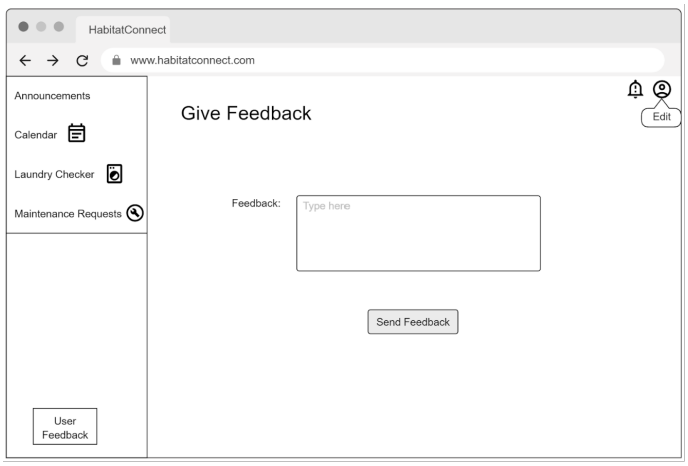

Fig. 30: User feedback page

When the submit button is clicked, the feedback will automatically be relayed in email form to website developers.

I. User profile options

In the user profile section, users have the flexibility to make adjustments to several key aspects of their accounts to ensure a personalized experience. They are able to modify the birthday (in case of an initial mistake), update the username to reflect a new online identity, and change the profile picture to express their personality. To find user profile options, users must look at the top right of the website with the profile icon, and click "Edit" in order to end up on the profile screen.

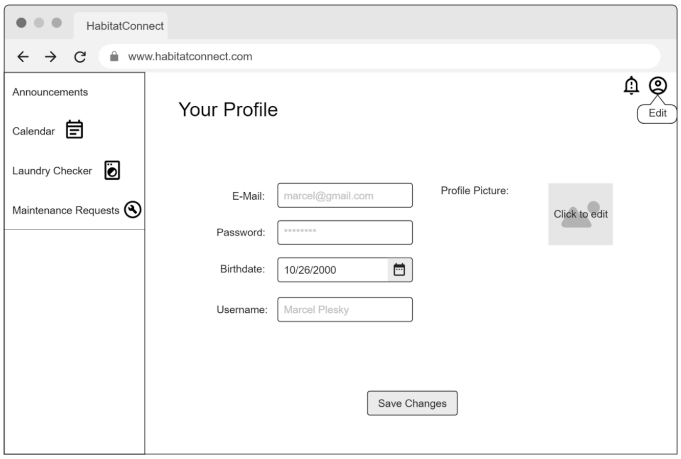

Fig. 31: Profile page

CHANGE PASSWORD

The user can change their passwords as many times as they want. However, passwords cannot be repeated. They must also follow the original protocols regarding the required elements of passwords. Upon changing the password, a confirmation email will be sent to the user.

CHANGE BIRTHDAY

Upon initially signing up for our web applications, users can commonly mistakenly put down the wrong birthday. However our feature allows users to correct that mistake. This feature can be used infinite times.

CHANGE USERNAME

Within the user profile, users can retain the ability to modify their username to better suit their identity within Habitat Connect. To enhance freedom of expression, users will have limited restrictions upon choosing their usernames. Restrictions include:

- `Username length`: string of maximum 11 characters
- `Special characters`: 1 special character allowed
- `Duplication`: string cannot be identical to another username in use

CHANGE PROFILE PICTURE

If users want to further showcase their personality, they can add/edit their profile picture so that other users can see. There will be no restrictions for this feature, and users can change their profile pictures as much as they want.

V. ARCHITECTURE DESIGN & IMPLEMENTATION

A. Overall Architecture

In this subsection, we will provide a detailed description of the overall system architecture. In the figure below, we can see the relationship between...

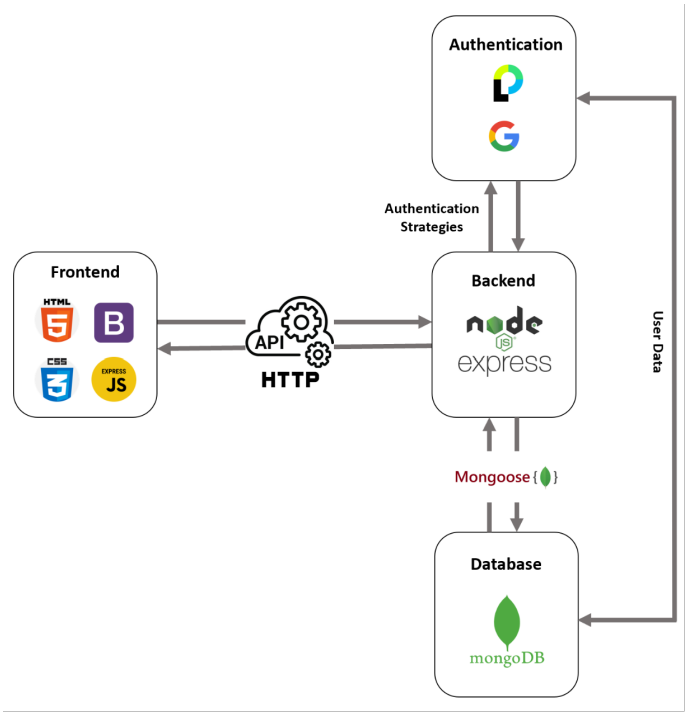

Fig. 32: Architecture schema

B. Directory Organization

By organizing our directory structure, we can maintain a clean and scalable codebase. We have defined a clear separation between frontend and backend components, allowing us to enhance code readability while streamlining the development and collaboration. Our overall directory structure is:

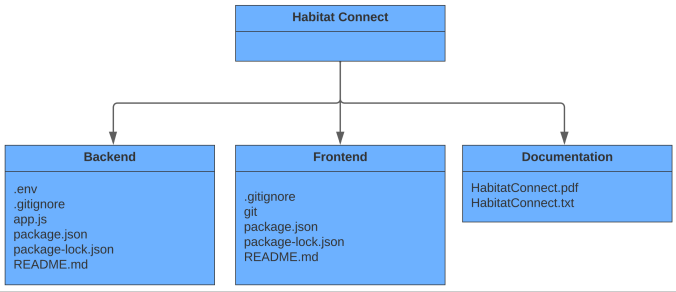

Fig. 33: Overall directory organization

MODULE 1: FRONTEND

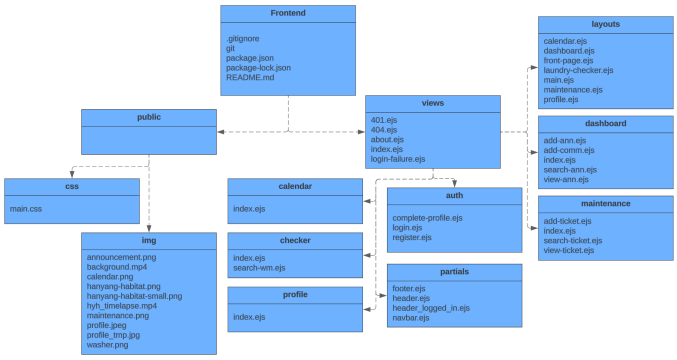

Fig. 34: Frontend module (See Table IV)

1) PURPOSE: The frontend module is our project's user-facing interface, which is intended to be engaging and intuitive. The purpose of the frontend is enabling possibilities for creating announcements and repair requests, as well as real-time monitoring of washing machine statuses and more. This module ensures a seamless user experience by presenting data fetched from the backend, with client-side validation delivering quick feedback. It connects to backend APIs to do tasks including creating, updating, and deleting announcements and maintenance requests. The interface is responsive and not dependent on any specific devices, with a component-based architecture for modular development and scalability. The use of specific visual components improves the entire user experience by making interactions with announcements, calendar, maintenance requests, washing machine status, and profile setting more dynamic and interesting.
2) FUNCTIONALITY: The frontend functionality is crafted using a combination of HTML, CSS, and JavaScript, providing a solid foundation for the user interface. HTML structures the content, while CSS adds style, ensuring a visually appealing design. JavaScript, along with the EJS templating engine, provides dynamic content updates and interactivity. Bootstrap, a powerful CSS framework, enhances responsiveness and streamlines the design process with pre-built components. The integration of these technologies results in a maintainable structure, enabling efficient development. The application's frontend leverages AJAX for asynchronous data retrieval, ensuring a seamless and responsive user experience. Furthermore, responsive design principles are implemented to guarantee compatibility across various devices, making it easy for users to create announcements, submit maintenance requests, and more functionalities effortlessly.
3) LOCATION OF SOURCE CODE: Can be found in `/HabitatConnect/Frontend`
4) CLASS COMPONENTS:
   a) General pages
      - 401.ejs: this page represents an Error 401 - Unauthorized response, indicating that the user lacks the necessary credentials to access a specific resource. It displays an error message and an explanatory paragraph. Users are guided to navigate back to the main page through a "Back to Main" button
      - 404.ejs: this page presents an Error 404 - Not Found message, indicating that the requested resource is not available. It displays an error message along with a descriptive paragraph. Users are guided to either check the URL or easily navigate back to the homepage using the "Back to Main" button
      - about.ejs: this is the "About" page for our project. It is intended to use it at the moment but it will be prepared for future purposes. It will contain information about the team and the goal of this project
      - index.ejs: this is the landing page of our website. Since it is the first page that the user sees it contains the login and register buttons. The user gets greeted with a welcome message and a timelapse video of Wangsimni is running in the background
      - login-failure.ejs: the user gets redirected to this error page in case the login fails. It shows a message with possible causes for the failed login and provides a button to redirect the user back to the login
   b) public/css & public/img
      - main.css: styling sheet for all the frontend components of this project
      - /img: here are all images and videos stored that are being displayed in the user interface
   c) views/calendar
      - index.ejs: this page represents a calendar. It uses CSS styles to structure and format the calendar layout. The calendar dynamically generates date boxes using EJS and Node.js, highlighting birthdays based on user data. JavaScript functions enable navigation to previous and next months, allowing users to interact with the calendar
   d) views/checker
      - index.ejs: this page features a table displaying information about each washing machine, including its name, current status, and options to change its status. If there are washing machines available, the table is populated dynamically based on the data received. Users can see the current status of each machine and choose to either mark it as "In Use" or "Finished" using corresponding buttons. These buttons trigger form submissions, updating the status of the selected washing machine. If there are no

washing machines available, a message with a small logo apologizes and suggests returning to the dashboard
- search-wm.ejs: this is a search-bar-component that serves as a search results display within the washer page. If search results are available, it shows a list of clickable washer-titles, each linking to the corresponding washer. If there are no search results, it displays a simple message indicating that no items were found.

e) views/profile
- index.ejs: the profile page provides users with the ability to view and update their profile information. The page welcomes the user by their username and includes a form for updating various details. If there are any validation errors, they are displayed at the top in a red alert box. The form includes fields for updating the username, room number, and birthdate. Additionally, users can upload or change their profile picture. If a profile image is already set, it is displayed; otherwise, a default image or placeholder is shown. The ”Update” button triggers the form submission for updating the profile

f) views/auth
- complete-profile.ejs: this page is a form for users to submit essential details required to finalize their profile. The form includes input fields for a username, room number, and birthdate. Error handling is implemented to display specific error messages if any issues arise during form submission
- login.ejs: this is the login page where the user can either login with an email/password combination or use the Google Sign-In feature
- register.ejs: this page is used to create a new user in the system using either an email/password combination or the Google Sign-In information. Afterwards the user gets forwarded to complete their profile details

g) views/partials
- footer.ejs: this is the footer element containing basic information about this project. The footer can be reused in the layouts for all pages
- header_logged_in.ejs: this is the header element including a title and search-bar. This specific header is only being displayed to logged in users. The difference is that instead of the login button there is a logout button. Reusable in all the layout pages
- header.ejs: this is the basic header element that is being shown to not logged in users, showing the login and register buttons. It is also missing the search-bar that is only displayed to logged in users. Used in the layout pages
- navbar.ejs: this is a vertical navigation bar that provides navigation buttons consisting of title and icon. The navigation bar is being reused in the layout pages as a component like the header and footer for example

h) views/maintenance
- add-ticket.ejs: this page serves as the submission form for maintenance requests, allowing users to submit tickets with a title and description. It includes form validation to ensure that the title and description do not exceed specified character limits (500 characters for the title and 1000 characters for the description). If the form submission is successful, a modal (”addModal”) appears to confirm the submission, providing information about the expected closure of the request within the next three days. In case of an error, such as empty fields or exceeding character limits, an error modal (”errorFormModal”) is displayed, guiding users to correct the form issues
- index.ejs: this page displays maintenance requests, providing users with information about their open requests that are currently being worked on. Users can create new maintenance requests using the ”Create new Request here” button if they don’t have any open requests. For users with open requests, the page shows a table with the title and a truncated description of each request, along with an option to view/update each request. The table includes pagination for easy navigation through multiple requests. If a user doesn’t have any open requests, a message encourages them to create a new request using the provided button
- search-ticket.ejs: this is a search-bar-component that serves as a search results display within the maintenance requests. If search results are available, it shows a list of clickable titles, each linking to the detailed view of the corresponding requests. If there are no search results, it displays a simple message indicating that no items were found
- view-ticket.ejs: this page allows users to view and update a specific maintenance request. The navigation provides a link back to the main Maintenance Request page and displays the title of the current ticket. The page includes a form for updating the ticket’s title and description. Users can submit changes using the ”Update” button, and there’s an option to close the ticket using the ”Close” button, which triggers a confirmation modal. If there’s an error, it will be displayed at the top of the page in a red alert box

i) views/layouts

- calendar.ejs, dashboard.ejs, front-page.ejs, laundry-checker.ejs, main.ejs, maintenance.ejs, profile.ejs: the layout structure of this project is designed for modularity and consistency across various views. Each layout follows an HTML structure, incorporating EJS for dynamic content rendering. The ¡head¿ section includes dynamic metadata, such as the page title and description. External styles are applied through Bootstrap v5.3.2 and a project-specific stylesheet. Google Fonts provides more design-choices. The layout is organized into sections: header, navbar (or sidebar), main content area, and footer, allowing for the inclusion of dynamic content through EJS partials. This approach promotes code reusability and a unified user interface with the ability to adjust each page's layout easily

j) views/dashboard

- add-ann.ejs: this page displays a form to add a new announcement to the announcement board. The form includes a title and textbody that gets sent to the backend and validated
- add-comm.ejs : users are able to post comments under a posted announcement. This page includes a form with a textbox that sends the comment to the backend and gets validated
- index.ejs: this page is a dynamic dashboard displaying announcements. It starts with a greeting to the user and offers a "Create" button to add new announcements. The main content area shows a grid of announcements, each represented by a bootstrap card with a title and a preview of the content. If there are no announcements, it prompts the user to create their first announcement with a call-to-action link. The page includes a pagination feature at the bottom to navigate through the announcements efficiently
- search-ann.ejs: this is a search-bar-component that serves as a search results display within the dashboard. If search results are available, it shows a list of clickable titles, each linking to the detailed view of the corresponding announcement. If there are no search results, it displays a simple message indicating that no items were found
- view-ann.ejs: this page serves as the view for a specific announcement within the dashboard. It displays the announcement's title, body, and creator, and provides functionality for the announcement owner to update or delete the announcement. Users who are not the announcement owner can only view the announcement details. Additionally, the page includes a comments section, allowing users to view existing comments and, if they are the comment creator, update or delete their comments

5) How and why we use it? HTML, CSS, Bootstrap, JavaScript, and EJS form a robust frontend toolkit. HTML structures content, CSS styles layouts, and Bootstrap ensures consistency. JavaScript adds interactivity, enabling dynamic features. EJS, integrated with Node.js, dynamically generates HTML based on server-side data. Since we have Android and iOS users in our group we decided against developing a mobile application and instead chose to focus on a web-application that can provide us with new programming experiences and help us in future projects.

Module 2: Backend

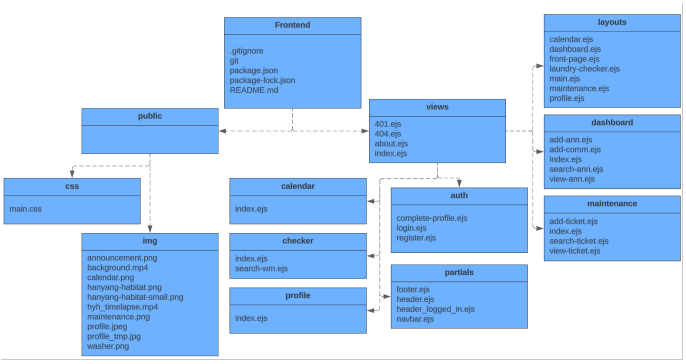

Fig. 35: Backend module (See Table V)

1) Purpose: The backend module forms the backbone of our project, focusing on server-side operations to handle requests from the frontend. It plays a pivotal role in managing and maintaining data integrity for user profiles, announcements, maintenance requests, and washing machine statuses. Through robust APIs, it facilitates seamless communication with the database, ensuring secure and scalable functionality. This backend system is designed to support critical processes such as user authentication, data storage, and retrieval, laying the foundation for a dynamic and engaging user experience crafted by the frontend module.
2) Functionality: The backend module's functionality is made possible using Node.js, Express.js and Mongoose to handle server-side logic and data management. Its primary purpose is to process requests from the frontend, providing seamless communication with the database. Node.js ensures an architecture for efficient handling of concurrent operations. Express.js simplifies route handling and streamlining the development of APIs. Mongoose, a MongoDB object modeling tool, provides an elegant interface for interacting with our database, enhancing data integrity and retrieval. This backend system supports user authentication, manages

data persistence for user profiles, announcements, maintenance requests, and washing machine statuses. By using these technologies, we achieve a scalable, efficient, and secure backend that complements the frontend's dynamic user experience.

3) LOCATION OF SOURCE CODE: Can be found in `/HabitatConnect/Backend`
4) CLASS COMPONENTS:
   a) Environment settings
      - .env: this configuration file holds key-value pairs crucial for our application's functionality. It stores sensitive information such as the MongoDB URI, which connects our backend to the database hosted on MongoDB Atlas. The Google Client ID and Secret are essential for enabling Google OAuth, ensuring secure user authentication. The specified callback URLs, both for registration and login, are important for handling the redirection and authentication processes when users sign up or log in using their Google accounts
      - app.js: the app.js file serves as the entry point for our Node.js Express application, managing various components. It employs middleware like express-ejs-layouts, method-override, and express-flash to enhance routing and manage HTTP methods effectively. The configuration includes sessions, utilizing express-session with a MongoDB store for persistent user data. Passport.js is integrated for authentication, enabling secure user management. Routes are organized and directed to respective controllers. The templating engine EJS is employed to structure dynamic views. Connection to the MongoDB database is established through the connectDB function. The file is structured to handle various routes, such as authentication, index, calendar, dashboard, laundry checker, maintenance, and profile. It also ensures a 404 response for any unrecognized paths. The application listens on the specified port
   b) server/config
      - db.js: this config file uses Mongoose to connect a Node.js application to a MongoDB database specified by the "MONGODB_URI" environment variable. This module is crucial for maintaining a connection to the database, ensuring seamless data operations within the backend
   c) server/middleware
      - checkAuth.js: this middleware component checks whether the user is authenticated. If the user is authenticated, it allows the request to proceed to the next route handler. If the user is not authenticated, it renders a "401 Unauthorized" view, indicating that the user lacks the necessary credentials to access the requested resource
   d) server/routes
      - auth.js, calendar.js, dashboard.js, index.js, laundryChecker.js, maintenance.js, profile.js: all these routes are part of the system, providing the necessary endpoints like for example: user registration, login, dashboard access, maintenance ticket creation and many more. The actual implementation details are in the respective Controller module, which contains the logic for each route
   e) server/controllers
      - authController.js: this controller handles user registration, login, and profile completion. It supports both local and Google-based registration and login. For local registration, users provide an email and password, which is securely hashed and stored in the database. Google registration and login use OAuth2.0, allowing users to sign in with their Google accounts. The system also ensures a complete user profile with validation for username, room number, and birthdate. Users can update their profiles, and errors are displayed when validation fails
      - calendarController.js: this controller provides user birthdays and the current date. User data, including user IDs, usernames, and birthdates, is fetched from the database
      - checkerController.js: this controller manages the logic related to washing machines, such as retrieving their status, updating status based on user input, and handling search functionality. Additionally it controls the amount of washing machines that should be shown per page
      - dashboardController.js: this controller, associated with announcements and comments in the Dashboard page, handles views for displaying, updating, and deleting announcements. Users can add comments, update them and perform searches. The controller ensures length validation for titles, bodies, and comments, redirecting appropriately after each operation
      - mainController.js: this controller manages views for the homepage of the web application. The homepage displays the main content with a layout for the front page
      - maintenanceController.js: this controller manages maintenance-related functionalities, including viewing, updating, deleting, adding and searching tickets. The "maintenance" view displays a paginated list of maintenance tickets for the user, with options to view, update, and delete each ticket. Users can also add new maintenance tickets, search for specific tickets, and view detailed information about a specific ticket. The

controller ensures that users can only access and modify their own tickets
- profileController.js: this controller handles profile-related functionalities. The ”profile” view displays the user’s profile information, including username, room number and birthdate. Users can update their profiles by submitting the provided form, and the controller validates the input data, ensuring that the username is within a certain character limit, the room number is within a specified range and the birthdate is in a valid format and not in the future. If there are errors in the input data, the controller renders the profile view again with error messages. If the update is successful, the user is redirected to the profile page

f) server/models
All the models are designed to interact with the appropriate MongoDB collection in a Node.js application using Mongoose.

- Announcements.js: this model represents announcements in a MongoDB database. It includes fields for the announcement’s owner, title, body, creation date, last update date, and an array of references to associated comments

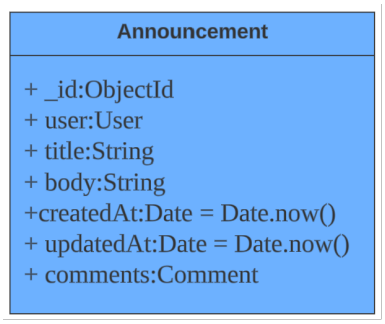

Fig. 36: Announcement schema

- Comment.js: this model represents announcement comments. It includes fields for the ticket’s owner, title, body, creation date, and last update date

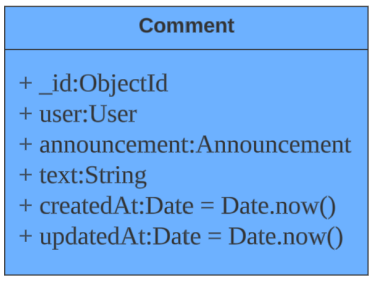

Fig. 37: Comment schema

- Ticket.js: this model represents maintenance tickets. It includes fields for the ticket’s owner, title, body, creation date, and last update date. Additionally, the model is configured with an index to automatically expire tickets after 3 days, enhancing data management

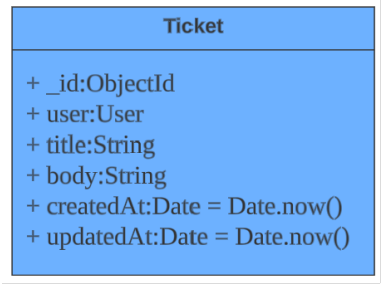

Fig. 38: Ticket schema

- User.js: this model includes fields such as email, password, googleId, username, profileImage, birthdate, and roomNumber. Each field has specific validation criteria. The model is designed to manage user information, providing a structured schema for registration, authentication, and profile management within the application

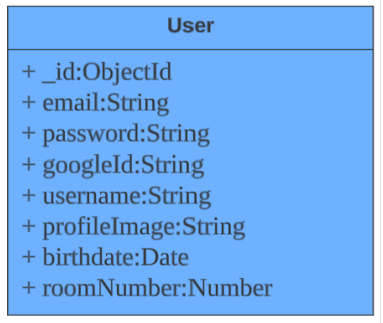

Fig. 39: User schema

- WashingMachine.js: this model represents the status of a washing machine. It includes fields for “status” and “updatedAt”. The “status” field has possible values of “Available” and “In use”, with a default value of “Available”. This model is designed to manage and track the status of washing machines in the application

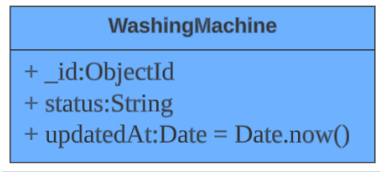

Fig. 40: Washing Machine schema

5) WHERE IS IT TAKEN FROM?
6) HOW AND WHY WE USE IT? We chose this backend technology because it aligns with modern web development practices, emphasizing scalability and maintainability. In Visual Studio Code, we use Node.js and Express for development. This helps us write and organize code easily. Opting for Node.js and MongoDB helps us gain valuable experience with these relevant technologies for future projects.

MODULE 3: DATABASE - MONGODB

1) PURPOSE: MongoDB serves as the primary database solution for our application, providing a flexible and scalable storage system for managing various data types.
2) FUNCTIONALITY: MongoDB is a NoSQL database that excels in handling unstructured and semi-structured data. It supports a document-oriented data model, allowing us to store and retrieve complex data structures with ease.
3) LOCATION OF SOURCE CODE: Can be found in `/HabitatConnect/Backend`
4) CLASS COMPONENTS: The components consist of schemas and models that define the structure and behavior of the data stored in MongoDB. These components include schemas for users, announcements, comments, tickets, and washing machines like mentioned in the backend part.
5) WHERE IS IT TAKEN FROM? MongoDB is an open-source, document-oriented database system. We utilize the official MongoDB-Node.js driver and Mongoose, an ODM (Object Data Modeling) library for MongoDB and Node.js, to interact with the database.
6) HOW AND WHY WE USE IT? We chose MongoDB for our database because it's flexible and can handle a lot of different types of data. This is important for us because our data needs might change over time. MongoDB's way of storing data is similar to how JavaScript works, which makes it easy to connect with our Node.js backend. To manage documents directly we use the cloud based MongoDB Atlas where everything can be configured. Using MongoDB fits well with our goal of creating a modern and scalable web app, making sure users have a smooth and responsive experience.

VI. USE CASES

In this section, we will provide a detailed description of some use cases that demonstrate all requirements implemented. Some examples of questions we aim to answer are: how does a user register in Habitat Connect? What happens if the validation check is not successful? How can I user comment on an announcement? We will explore these and more practical scenarios in detail to get a better understanding of our software behavior.

A. Landing Page

1) Log In: if the user clicks the "Log In" button in the top right corner, the log in form will be shown. The user can either log in with an email/password combination or choose the "Sign in with Google" option via the button below.

AUXILIARY TABLES

Directory	File Name
/HabitatConnect/Frontend/	.gitignore git package.json package-lock.json README.md
/HabitatConnect/Frontend/public/css	main.css
/HabitatConnect/Frontend/public/img	announcement.png calendar.png hanyang-habitat.png hanyang-habitat-small.png hyh_timelapse.mp4 maintenance.png profile.jpeg profile_tmp.jpg washer.png
/HabitatConnect/Frontend/views	401.ejs 404.ejs about.ejs index.ejs
/HabitatConnect/Frontend/views/auth	complete-profile.ejs login.ejs register.ejs
/HabitatConnect/Frontend/views/calendar	index.ejs
/HabitatConnect/Frontend/views/checker	index.ejs search-wm.ejs
/HabitatConnect/Frontend/views/dashboard	add-ann.ejs add-comm.ejs index.ejs search-ann.ejs view-ann.ejs
/HabitatConnect/Frontend/views/layouts	calendar.ejs dashboard.ejs front-page.ejs laundry-checker.ejs main.ejs maintenance.ejs profile.ejs
/HabitatConnect/Frontend/views/maitenance	add-ticket.ejs index.ejs search-ticket.ejs view-ticket.ejs
/HabitatConnect/Frontend/views/partials	footer.ejs header.ejs header_logged_in.ejs navbar.ejs
/HabitatConnect/Frontend/views/profile	index.ejs

TABLE IV: Frontend Module Table

Directory	File Name
/HabitatConnect/Backend/	.env .gitignore app.js package.json package-lock.json README.md
/HabitatConnect/Backend/node_modules	(Node.js Project Dependencies)
/HabitatConnect/Backend/server/config	db.js
/HabitatConnect/Backend/server/controllers	authController.js calendarController.js checkerController.js dashboardController.js mainController.js maintenanceController.js profileController.js
/HabitatConnect/Backend/server/middleware	checkAuth.js
/HabitatConnect/Backend/server/models	Announcements.js Comment.js Ticket.js User.js WashingMachine.js
/HabitatConnect/Backend/server/routes	auth.js calendar.js dashboard.js index.js laundryChecker.js maintenance.js profile.js

TABLE V: Backend Module Table

REFERENCES

[1] Facebook for Developers, "WhatsApp Business API - Documentation," Facebook for Developers. [Online]. Available: https://developers.facebook.com/docs/whatsapp/

[2] "KakaoTalk - Services," Kakao Corporation. [Online]. Available: https://www.kakaocorp.com/page/service/service/KakaoTalk?lang=en

[3] "GitHub - Discord," GitHub. [Online]. Available: https://github.com/discord

[4] "Google Calendar API Documentation," Google Developers. [Online]. Available: https://developers.google.com/calendar

[5] "CI Hackathon July 2020 - GitHub Repository," GitHub. [Online]. Available: https://github.com/maliahavlicek/ci_hackathon_july_2020

[6] "Zendesk - GitHub Repository," GitHub. [Online]. Available: https://github.com/zendesk

[7] "Svelte - GitHub Repository," GitHub. [Online]. Available: https://github.com/sveltejs/svelte

[8] "Node.js - GitHub Repository," GitHub. [Online]. Available: https://github.com/nodejs

[9] "Node.js - Documentation," Node.js. [Online]. Available: https://nodejs.org/en/docs

[10] "Express.js Documentation," Express.js. [Online]. Available: https://expressjs.com/

[11] "Bootstrap - GitHub Repository," GitHub. [Online]. Available: https://github.com/twbs/bootstrap

[12] "Axios - GitHub Repository," GitHub. [Online]. Available: https://github.com/axios/axios

[13] "MongoDB," MongoDB. [Online]. Available: https://www.mongodb.com

[14] "MongoDB - Documentation," MongoDB. [Online]. Available: https://www.mongodb.com/docs/manual/

[15] "Visual Studio Code - GitHub Repository," GitHub. [Online]. Available: https://github.com/microsoft/vscode

[16] "Git," Official Git Website. [Online]. Available: https://git-scm.com/

[17] "Overleaf - Documentation," Overleaf. [Online]. Available: https://www.overleaf.com/learn

[18] "Moqups - Online Moqups, Wireframes & Design Tool," Moqups. [Online]. Available: https://moqups.com/

[19] "JavaScript Documentation," Mozilla Developer Network (MDN). [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript

[20] "HTML Documentation," Mozilla Developer Network (MDN). [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTML

[21] "CSS Documentation," Mozilla Developer Network (MDN). [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/CSS]