Software Design Document for I.Helpr

Project Name: I.Helpr

Ji Won Choi	student # 111230163	jiwon.choi.2@stonybrook.edu
Sujeong Youn	student # 111290244	sujeong.youn@stonybrook.edu
Pyungkang Hong	student # 110054399	pyungkang.hong@stonybrook.edu

Course:	CSE 416
Date:	10/17/2020

Table of Contents

1. Introduction

1.1 Product description

In this day and age, the number of dual-income households is increasing, and many parents are seeking childcare services. Parents want a guardian whom they can trust. So, they either ask around to find someone within the circle of people that they know or find one through agencies. These methods require unnecessary time and effort if the preferences between the guardian and parents are not met. Especially, finding a babysitter through an agency involves the mediation of every conversation between parents and caregivers, which makes it time-consuming. Moreover, this process is not only time consuming and costs money, but also uncertain for parents whether they can trust the caregivers sent from the agency by the name value of the brand. To alleviate all this problem, we are building a web application that can provide the following features:

i) Parents can choose a sitter from a larger pool of people with few mouse clicks :

Parents will be able to find the most suitable sitter by posting a job opening with the conditions stated or by looking through a list of sitters' job hunting posts. When a job opening post is posted, there will be sitters applying for the job or sending messages regarding the position. Among the list of applicants, parents can decide whom they want to hire and send a message to decide when to start the work and talk about the work details.

ii) A grading system and review will ensure the credibility of the employee :

Parents will be able to rate the guardians and leave a review of the work that they did after their work is done. Thus, other parents can refer to the reviews and ratings before they decide which caregiver they will hire.

iii) The list of job postings can save the time needed for the sitter to find the most complete babysitter because of all the conditions entered in each post.

It can be annoying to go through all the applications on the web page.To make the situation easier, parents will be able to enter correct inputs (selecting options from drop-down lists or entering specific information) so that they can match applications with the conditions they require.

iv) Messaging system between parents and caregivers :

As mentioned in the introduction the whole process of finding a sitter is time-consuming. This application will provide a messaging system to enable the direct conversation between the parents and caregivers.

With the childcare service application, parents can conveniently search and evaluate caregivers for their children from a website, instead of needing to go through the challenges coordinating with agencies or trying to enlist their friends. The development of this web app will not only facilitate the parents searching for caregivers but also it will open up opportunities for potential caregivers to work and make extra income.

1.2 Scope

Our web application will provide a service that facilitates the process of parents finding their caregivers and caregivers finding a job. This service will be usable for any parents and caregivers. Our initial aim is to provide service solely in Songdo and we plan to gradually extend the service area to South Korea as the service stabilizes. Aside from our plan to provide a virtual space for parents and caregivers to satisfy their needs of finding a sitter or a job, there are certain things that we plan not to do. The application will not receive any fees or interests for service usage. As a service provider, we will not interfere with any matching process so that the users can decide whom they want to work with. The service will be a cross-platform web application meaning it will run on different operating systems. It will support any devices that can use the web such as smartphones, tablet PCs, and Laptops, etc.

1.3 Users

We classify our users into two groups--parent and sitter users. Although our application will be used by parents and anyone who would like to work as a sitter, we expect our users to have the features listed below:

i) Parent users

- Dual income parents
- Physically disable parents
- The parents with many children

ii) Sitter users

- People who love children
- A college student
- A person with child care certificate
- A housewives without a salary
- Experienced babysitters

The group of users we intend to use are people of all ages who understand and use our website, regardless of educational level.

1.4 User feedback

To build an application that provides convenient service, getting feedback is necessary during a process of application development. It helps the developers to get insights from people outside the team members. This process enables the developers to find possible improvements that they cannot think of. We think this feedback is most needed before planning the software design description (SDD) and after building the first demo of our project. If the feedback suggests to either add or delete a function, then implementing the change would be much easier when it is before considering the details of how the application operates. After building the first demo, getting feedback to find out whether the application works without an error would be crucial.

Recently our team got a chance to present our project to our classmates and we have received some feedback from them. The feedback mostly focused on how to guarantee the credibility of the sitters, especially when the sitter is a new user, and security and privacy issues. This process led us to concentrate on ways to increase the credibility of the sitters and not to reveal the users' personal information that they did not consent to.

1.5 Existing alternatives

Some companies provide direct labor to match people. There are also other web-apps like Sitternet and MomSitter which enable people to find a person who meets the condition.

Strengths:

- An ability for parents and sitters to filter and quickly match the person they want
- The ability for parents can share information about sitters. Sitters also have the ability to choose the area they want.
- Simple and comprehensive sign-up process
- Simple UI design which makes the web application easy to use.
- Provide the service to all regions in a country

Weaknesses:

- Requires payment to apply for a job

These existing alternatives have influenced our product and feature decisions as below:

- We are incorporating a simple UI design and comprehensive sign-up process
- We are going to provide the service without charging a commission

1.6 Definitions

- Web application: An application software that runs on a web server, unlike computer-based software programs that are stored locally on the Operating System of the device.
- UI design: User interface design or user interface engineering is the design of user interfaces for machines and software
- Understandability: Understandability expresses the clarity of the system design.

1.7 References

Babysits. “Oppas - Oppaswerk | Oppassen.Nl (Babysits).” Babysits, www.babysits.com.

Care.com. “Care.Com: Find Child Care, Senior Care, Pet Care and Housekeeping.” Care.Com, www.care.com.

“Jjaekkag-ag-eo.” 째깍악어 | 필요한 도움이 다 있다, www.tictoccroc.com.

“Dandihelpeo.” 단디헬퍼, www.dandihelper.com.

“mamsiteo.” 맘시터, www.mom-sitter.com.

“Sittercity: Find Local Child Care, Senior Care, & Pet Care.” Sittercity, www.sittercity.com.

2. Requirements

2.1 Functional Requirements

1. Home page

- Users can sign up for the website and they should choose whether to register as a sitter user or as a parent user

- Users can log in to the website

- Users can confirm ID and reset the password when they forgot it

2. My page

- Users can edit and view their personal information such as contact number, email address, date of birth, and password

- Users can view the previous messages they sent and received

- Users can check their average ratings and reviews on my page

- Among the people who they sent messages to, users can leave reviews and ratings and the user should indicate whether they had actually worked with that person (Sitter users can leave reviews and ratings to parent users and parent users can leave review and ratings to sitter users)

- Sitter users are able to store their application in their profile to make it easier to apply to jobs

- Sitter users can view, delete or edit their job-wanted posts that they previously posted

- Sitter users can view the applied records and they can also cancel applying

- Parent users can view, delete or edit their job openings that they previously posted

- Parent users can view the list of applicants on the job opening page that they previously posted

3. For parents page

- Sitter users can post a job-wanted post with the work conditions such as how much they want to get paid(pay rate), how long they can work(work period), what time they can work(work time), and available workplace(location).

- Parent users can search sitters with title

- Parent users can also search sitters with matching filters such as work period, work time, desired gender of applicants, location, and pay rate

- If the parent user wants to interview the sitter or talk about the work details, they can send a message.

- If the parent user wants to save the post and view it later, they can add the post to 'My favorite'

4. For sitters page

- Sitter users can search jobs (parents) with the post title

- Sitter users can also search jobs with matching filters such as pay rate, work period, work time, gender and location.

- If the sitter user has any questions or wants to talk about the work details, they can send a message.

- Sitter users can apply for the job by clicking the apply button and they can fill out the application form or bring the application that the sitter had written before.

- If the sitter user wants to save the post and view it later, they can add the post to 'My favorite'

- Parent users can post a job opening with the conditions that they are looking for. Such as how much they can pay for a sitter(pay rate), how long they need a sitter(work period), what time they need a sitter to take care of their children(work time), gender and location.

5. Information page

- Users can be informed how the website works and get the information of this website

- Users can view the frequently asked questions and answers

2.2 Use case diagram

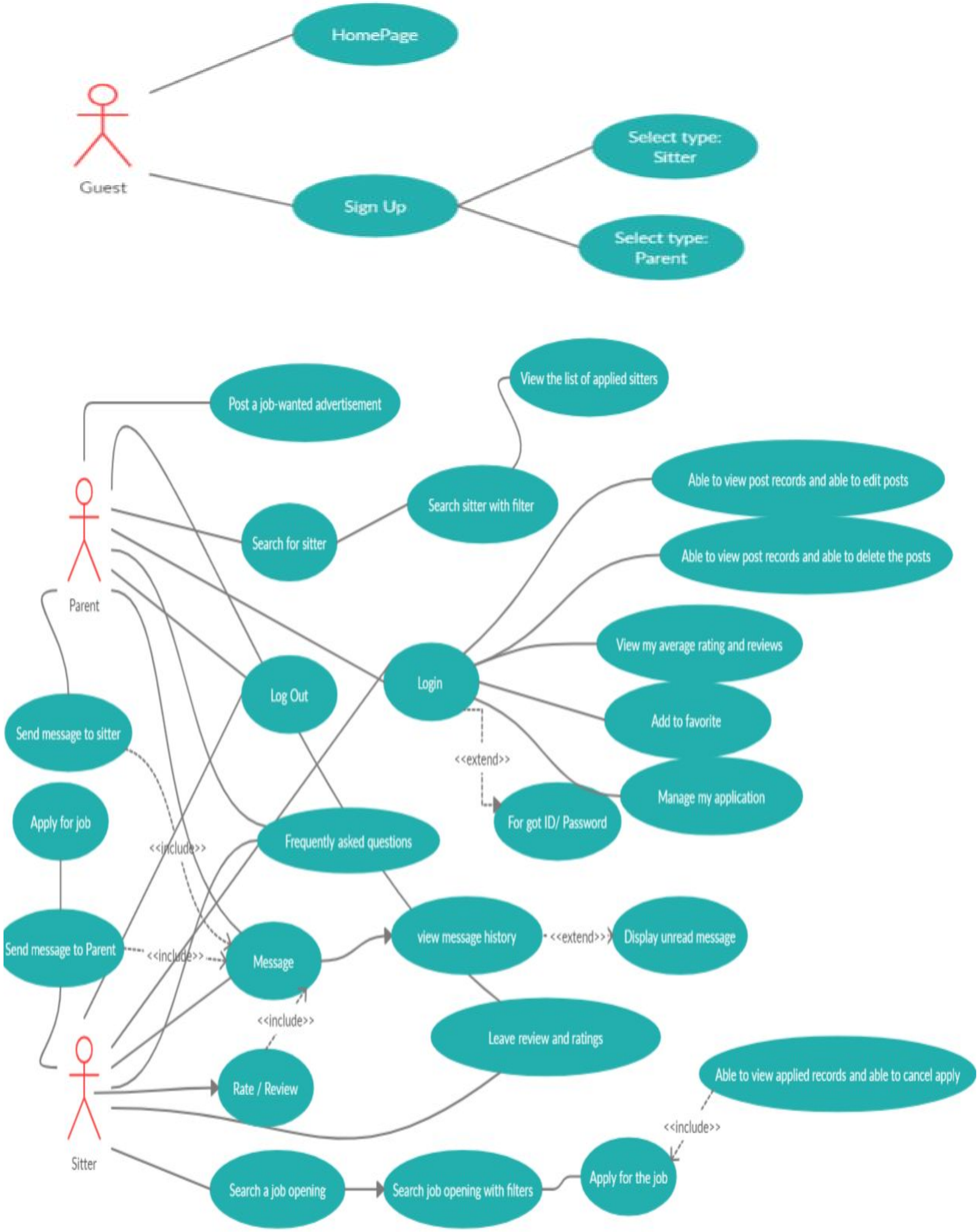


Figure 1. Use case diagram

2.3 Use cases

Use Case 1: Sign up

Use-case:	Sign up
Primary actor:	All Users
Goal in context:	The user can sign up for the web app.
Preconditions:	The user is using the web app.
Scenario:	1. User clicks the 'Sign up' button. 2. User should choose whether to register as a sitter or a parent 3. If the user selects a sitter, they have to put their name, ID, password, date of birth, email, contact number, and gender. If the user selects a parent user, they have to put their name, ID, ID check password, email, children info, and contact number. 4. After filling out personal information, click the 'Done' button.
Extensions:	Users are not able to sign up again with the same email. If the user didn't fill out all the required information, then they will be asked to fill out all the information.
Priority:	Essential

Use Case 2: Forgot Password

Use-case:	Forgot Password
Primary actor:	All Users
Goal in context:	The user can retrieve their password/id.
Preconditions:	The user is logged in as a parent user.
Scenario:	1. User starts the application. 2. User clicks the 'Forgot Password' button. 3. The user is prompted to verify his or her information about the id and able to retrieve the password.
Extensions:	Users may not be able to retrieve their password or id if they cannot verify their identity.
Priority:	Expected

Use Case 3: Post a job opening

Use-case:	Post a job opening
Primary actor:	Parent Users
Goal in context:	The parent user can post a job opening on the ‘For sitter’ page.
Preconditions:	The user is logged in as a parent user.
Scenario:	1. User clicks the 'Create Job Opening' button on the ‘For sitter’ page. 2. The user is prompted to select the work conditions such as pay rate, work period, work time, location, what kind of service they want, and gender. 3. The user is also prompted to write the title and contents of the job opening. 4. User clicks the 'Post' button after they finish writing.
Extensions:	'Create Job Opening' button is always there, but if a user has not logged in as a parent user, they will not be able to post a job opening.
Priority:	Essential

Use Case 4: Search for a sitter

Use-case:	Search for a sitter
Primary actor:	Parent Users
Goal in context:	The parent user can search sitters with the title
Preconditions:	The user is on the ‘For parents’ page.
Scenario:	1. User types in the search words (post title) in the search engine. 2. User clicks the 'Search' button. 3. The user can see the result containing the words they typed in.
Extensions:	If there is no result, then they will be notified with text that says 'No result found.' If a user searches without typing anything, it will display all the posts.
Priority:	Essential

Use Case 5: Search sitters with filters

Use-case:	Search for a sitter with filters
Primary actor:	Parent Users
Goal in context:	The user can search sitters with filters such as pay rate, location,work time, work period, what kind of work they want, and gender.
Preconditions:	The user is on the 'For parents' page.
Scenario:	1. Parent user chooses conditions such as pay rate, location, work time, work period, what kind of work they want, and gender. 2. After the user chooses all the conditions, they can click the 'Search' button. 3. List of sitters who match the conditions will show.
Extensions:	If there is no result, then they will be notified with text that says 'No result found.'
Priority:	Essential

Use Case 6: Sending messages to a sitter

Use-case:	Sending messages to a sitter
Primary actor:	Parent users
Goal in context:	Parent users can send messages to a sitter user.
Preconditions:	Parent users should be viewing a job-wanted post of a sitter they chose.
Scenario:	1. On a job-wanted post page of a sitter they chose, the parent user clicks the 'send a message' button. 2. It redirects to the 'write message page' so that the parent user can type in the message. 3. When the parent user clicks the 'send' button, the message will be sent to the sitter who posted the job-wanted post. 4. The sender can check the sent message on the message history session and the sitter who receives the message can check the message on the unread message session.
Extensions:	If a user clicks the 'send message' button without writing something, the user will get an alert to type something and the empty message won't be sent.
Priority:	Essential

Use Case 7: Sending messages to a parent

Use-case:	Sending messages to a parent
Primary actor:	Sitter users
Goal in context:	Sitter users can send messages to parent users.
Preconditions:	Sitter users should click the job opening post they are interested in and on that page, they can send a message.
Scenario:	This use case is the same as use case 6, except the 'Send a message' button is on a job opening post page and the role of sitter and parent are changed.
Extensions:	If a user clicks the 'send message' button without writing something, the user will get an alert to type something and the empty message won't be sent.
Priority:	Essential

Use Case 8: Sending messages while viewing the sent and received messages

Use-case:	Sending messages while viewing the sent and received messages
Primary actor:	Users
Goal in context:	Sitter users can send messages to parent users and parent users can send messages to sitter users while viewing the conversation.
Preconditions:	Sitter users should click the view button on the messages page.
Scenario:	1. On a message page the user clicks the view button and goes to view messages page. 2. After reading the previous messages, the user can write a message to send. 3. Click the send button.
Extensions:	If the user clicks the go back button while writing the message, the message won't be sent and will be returned to the messages page.
Priority:	Essential

Use Case 9: Apply for a job opening

Use-case:	Apply for a job opening
Primary actor:	Sitter users
Goal in context:	Sitter users can apply for a job opening.
Preconditions:	Sitter users should click the job opening post they want to apply and on that page, they can apply.
Scenario:	1. The sitter user clicks the 'Apply' button on a job-opening post that a sitter wants to apply to. 2. The user will be asked to fill out the form. 3. If there is any application that the user had written before, they can also select it from the select application page which will be shown when the 'Apply' button is clicked on the post. 4. On the select application page, the user will select an application which will be sent to the parent user who posted the job-opening post. 5. Clicking the 'Apply' button will show a pop up window that confirms the successful apply. 6. After the submission, the user will be redirected to mypage, where the user can see the applied history.
Extensions:	
Priority:	Essential

Use Case 10: View the list of applied sitters

Use-case:	View the list of applied sitters
Primary actor:	Parent users
Goal in context:	The parent user can view the list of applied sitters.
Preconditions:	Parent users are logged in and they are on 'My posts' page.
Scenario:	1. When parent users click the 'My posts' page on the link to 'Mypage', they can see the job openings they posted before. 2. The list will be shown in a table form. On a row, there is the post title, date, the 'Applicants' button and a 'Delete' button. 3. When the 'Applicants' button is clicked, the titles of the applications will appear which can be clicked to see the contents of the application.
Extensions:	If there are no applicants, no result will appear.
Priority:	Expected

Use Case 11: Able to view applied records and able to cancel applying

Use-case:	Able to view applied records and able to cancel applying
Primary actor:	Sitter users who applied to a job (job opening post).
Goal in context:	The user can view the list of job openings which they previously applied to and they can also cancel the application.
Preconditions:	The user should be logged in as a sitter user.
Scenario:	1. When a sitter user clicks 'Mypage', they can see the list of job openings that they previously applied to. 2. The user can click the 'Cancel' button and it will ask 'Are you sure you want to cancel the application?' 3. If a user clicks 'yes', the application will be canceled and removed from the list. If a user clicks 'cancel', the process will be cancelled.
Extensions:	If the user did not apply for any job openings, there should be no result in the 'Application records'.
Priority:	Essential

Use Case 12: Post a job-wanted advertisement

Use-case:	Post a job-wanted advertisement
Primary actor:	Sitter user
Goal in context:	The sitter user can post a job-wanted advertisement on the 'For parents' page.
Preconditions:	The user is logged in as a sitter user.
Scenario:	1. The user clicks the 'Create Job Wanted' button on the 'For parent' page. 2. The user is prompted to select conditions such as pay rate, work period, work time, location, and what kind of work they want. 3. The user is also prompted to write the title and the contents of the job-wanted post. 4. The user clicks the 'Post' button after they finish writing.
Extensions:	'Create Job Wanted' button is always there, but if a user did not log in as a sitter user, the user will not be able to post a job seeking post.
Priority:	Essential

Use Case 13: Search a job opening

Use-case:	Search a job opening
Primary actor:	Sitter users
Goal in context:	The sitter user can search jobs with a job opening title.
Preconditions:	The user is on the 'For sitters' page.
Scenario:	This use case scenario is the same as use case 4, except that it will show a list of job openings.
Extensions:	If there is no result, then they will be notified with text that says 'No result found.' If a user searches without typing anything, it will display all the posts.
Priority:	Essential

Use Case 14: Search job openings with filters

Use-case:	Search job openings with filters
Primary actor:	Sitter users
Goal in context:	The user can search jobs with filters such as pay rate, location, work time, work period, and what kind of work they want.
Preconditions:	The user is on the 'For sitters' page.
Scenario:	This use case scenario is the same as use case 5, except that it will show a list of job openings which satisfies the conditions.
Extensions:	If there is no result, they will be notified through the notification window.
Priority:	Essential

Use Case 15: Display unread messages

Use-case:	Display unread messages
Primary actor:	All Users
Goal in context:	The unread messages will be displayed separately so the user can notice it.
Preconditions:	The users are logged in to their account.
Scenario:	1. If the user clicks 'Messages' on my page, a list of unread messages will be displayed separately on top. 2. When a user clicks one of the unread messages, it will be deleted from the unread message list.
Extensions:	If there are no unread messages, there will be no result on the unread message list.
Priority:	Nice to have

Use Case 16: View message history

Use-case:	View message history
Primary actor:	All Users
Goal in context:	The user can view previous messages they sent and received.
Preconditions:	The users are logged in to their account.
Scenario:	1. If the user clicks 'Messages' in my page, the list of names that they communicated with will show up. 2. Users can view detailed messages they previously sent and received by clicking the view button.
Extensions:	If there are no messages they sent or received, there will be no result.
Priority:	Essential

Use Case 17: Message mark as read

Use-case:	Message mark as read
Primary actor:	All Users
Goal in context:	The user can mark the message as read without viewing
Preconditions:	The users are on the messages page and have unread messages.
Scenario:	1. If the user clicks ‘Messages’ in my page, the list of names that they communicated with will show up. 2. User clicks the 'Mark as read’ button. 3. The message which was in an unread message session will be marked as read and displayed on the message history session without reading the message.
Extensions:	
Priority:	Nice to have

Use Case 18: Delete message

Use-case:	Delete message
Primary actor:	All Users
Goal in context:	The user can delete the messages from the message history
Preconditions:	The users are on the messages page
Scenario:	1. If the user clicks ‘Messages’ in my page, the list of names that they communicated with will show up. 2. Users can delete the messages by clicking the 'Delete' button. 3. Users will be asked whether to delete the message. 4. If the user clicks yes, the message will be deleted and if the user clicks cancel, it will cancel the process.
Extensions:	
Priority:	Essential

Use Case 19: Leave reviews and ratings

Use-case:	Leave reviews and ratings
Primary actor:	All Users
Goal in context:	Users are able to leave reviews and ratings. (Sitter users are able to leave reviews and ratings to parent users and patent users are able to leave review and ratings to sitter users.)
Preconditions:	Users should be logged in and the user can only leave reviews and rates to the other user they have talked to(Messages).
Scenario:	1. If the user clicks ‘Messages’ in my page, the list of names that they communicated with will show up. 2. Users can click the 'Review' button 3. Users are redirected to the rate page, so the user can write a review and they can also rate between 1 to 5. 4. The user should also answer whether they had worked with the person. (Did you actually work with this person? Yes or No) 5. After answering, the user can click the 'Done' button and the review will be applied.
Extensions:	If the user didn’t write anything or didn’t select the rate, they will be alerted to fill out or to select the number. The review can be deleted if it contains swear words or if the contents are inappropriate.
Priority:	Essential

Use Case 20: Able to view post records and able to delete the posts

Use-case:	Able to view post records and able to delete the posts
Primary actor:	All Users
Goal in context:	The users can view the posts they previously uploaded and they are able to delete the posts. (The posts mean job openings for parent users and job-wanted posts for sitter users.)
Preconditions:	Users should be logged in
Scenario:	1. If the user clicks 'My posts' in my page, the list of titles of posts that the users uploaded previously will show up. 2. If a user clicks the 'Delete' button next to each title, an alert appears and asks 'Are you going to delete this post?' 3. If the user clicks 'yes', the post will be deleted and the applying history for the sitters who applied for the deleted post will be also removed.
Extensions:	If there are no posting records, nothing will be on the list.
Priority:	Essential

Use Case 21: Create application

Use-case:	Create my application
Primary actor:	Sitter users
Goal in context:	Sitter users can store their application in their profile to make it easier to apply to jobs and they can also view the previous applications.
Preconditions:	The user should be logged in as a sitter user.
Scenario:	1. User clicks my page and there is an 'Application' section where the user can see the application related activities that the user did. 2. If the user clicks the 'Create my application' button, the user can write a new application before they apply for a job. 3. A button click on 'Create my application' button will redirect the user to the 'Create my application' page where the user has to enter the user's information. 4. The user can click on the 'Apply' button when the user is done with filling in the application form.
Extensions:	If there is no application the user wrote before, the user will be asked to write a new application.
Priority:	Expected

Use Case 22: Delete application

Use-case:	Delete my application
Primary actor:	Sitter users
Goal in context:	Sitter users can view the previous applications and delete
Preconditions:	The user should be logged in as a sitter user.
Scenario:	1. User clicks my page and the user can see the application list. 2. If the user clicks the delete button, it asks "Are you sure to delete the application?" 3. If the user clicks yes, it will delete the application and if the user clicks cancel, the process will be canceled.
Extensions:	If the user applied to the job with the application that user deleted, the applying process will be canceled automatically.
Priority:	Expected

Use Case 23: View my average ratings and reviews

Use-case:	View my average ratings and reviews
Primary actor:	All Users
Goal in context:	Users can check their average ratings and reviews.
Preconditions:	Users should be logged in.
Scenario:	1. User is on my page. 2. Below the personal information, there is a 'My ratings' and 'My reviews' so the users can check it.
Extensions:	If there are no ratings, ratings will be set as 0/5 and "No reviews exist yet!" message will be displayed.
Priority:	Nice to have

Use Case 24: View the review history and able to delete

Use-case:	View the review history and able to delete
Primary actor:	All Users
Goal in context:	Users can check the ratings and reviews they have made and they can delete the review they made.
Preconditions:	Users should be logged in.
Scenario:	1. User is on my page. 2. There is a 'Review History' so the users can see the reviews they made. 3. They can delete the reviews and ratings by clicking delete and after the confirming message, it will be deleted.
Extensions:	If there are no reviews and ratings, "No reviews!" will be displayed.
Priority:	Nice to have

Use Case 25: Frequently asked questions

Use-case:	Frequently asked questions
Primary actor:	All Users
Goal in context:	The user can view the frequently asked questions.
Preconditions:	There are some questions that users frequently asked.
Scenario:	1. User clicks the 'Frequently asked' button. 2. It will redirect to the 'frequently asked questions' page. 3. There will be a list of frequently asked questions so the user can click one of them and view the answer for that question. 4. There is also an 'Ask' button so the user can click it and write the question there. 5. When the users click the 'Done' button, the question will be uploaded and the admin will answer for the question.
Extensions:	The question can be deleted if it contains swear words or if the contents are inappropriate.
Priority:	Nice to have

2.4 Non-functional Requirements

Verifiable

- The time spent to load any pages on the I.Helpr website should be less than 5 seconds in the minimum internet speed of 100Mbps.

Security

- The server must not return a restricted web page to any browser that it cannot authenticate.
- The server must not return a restricted web page to a user who is not authorized to access it.
- The application must not accept invalid URLs.
- The application must ensure the integrity of the customer account information with an embedded security system and authentication system in the django framework.

Understandability

- It must not take over 30 minutes for users who are new to the application when using the page.

3. System Architecture

3.1 Overview

The web application, I.Helpr will be built using Django 3.1, MySQL 8.0.21 for DB, Bootstrap 4, jQuery with Ajax. Since we plan to use Django, we will follow the MTV(Model Template View) pattern. For front-end programming, languages such as HTML, Javascript, CSS will be used, and python will be used for back-end. Because Django is a full-stack framework with a collection of python libraries, it does not require external libraries aside from web client libraries, such as urlparse, urllib2, and httplib, and a web server library, such as BaseHTTPServer.

We have used Bootstrap for our front-end design because it is easy to use, needless to build HTML and CSS from scratch, and it provides responsive HTML and CSS for smartphones, tablets, and desktops. We decided to use Bootstrap *version 4* because the components are mobile-first. This could be beneficial for us since we have to take our mobile users into account. Bootstrap is compatible with the most recent browsers such as IE10, Chrome, Firefox, Safari, and Opera, which reduces the risk of having broken CSS on different browsers.

As mentioned above, Django is a framework that is used to develop a complete web application, including front-end and back-end. It is used for developing a dynamic website, which is what we aim to build. Django is compatible with various cloud servers for deployments such as AWS, Google Cloud, Azure, etc. Because Django is already equipped with a security system that can protect the application from different attacks, it would guarantee a certain level of security. It provides common web application tools that our application needs such as messaging and authentication systems, etc. which will reduce the number of tasks that need to be done to build such functions.

3.2 UML Class Diagram

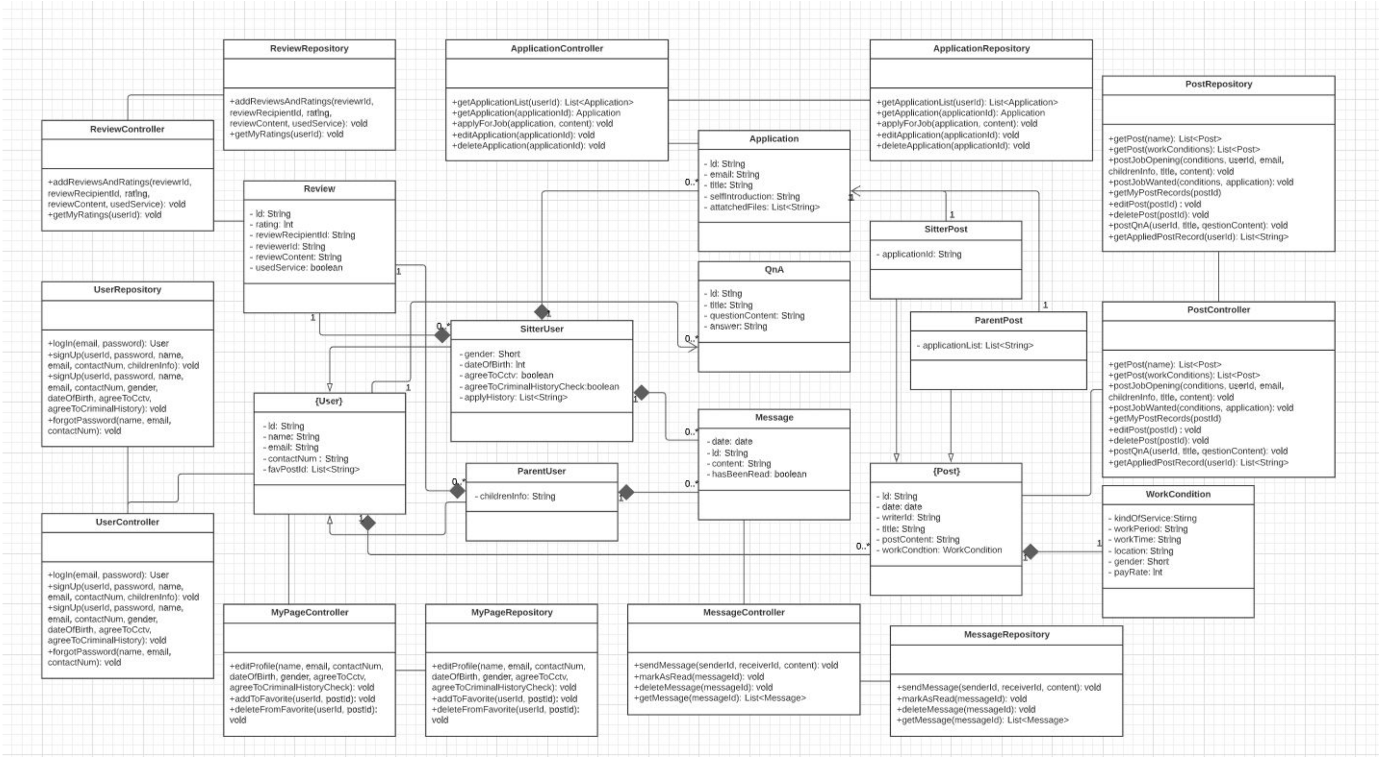
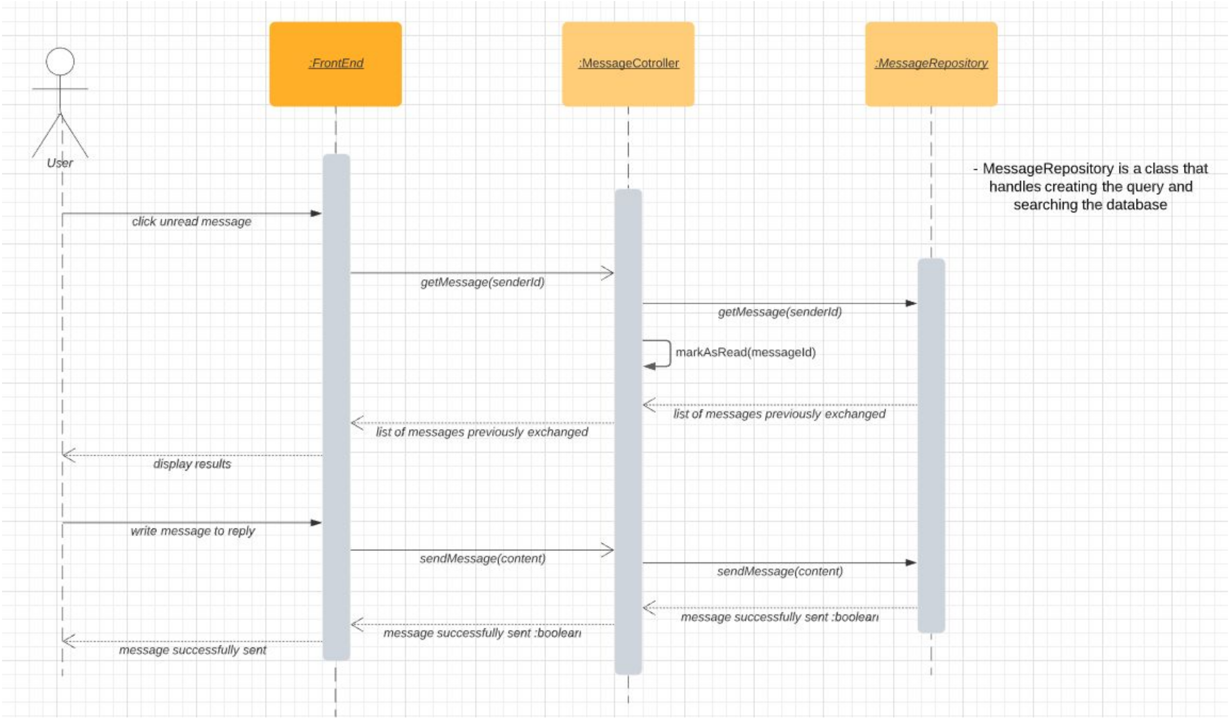

ReviewRepository
+addReviewsAndRatings(reviewrId, reviewRecipientId, rating, reviewContent, usedService): void
+getMyRatings(userId): void
ReviewController
+addReviewsAndRatings(reviewrId, reviewRecipientId, rating, reviewContent, usedService): void
+getMyRatings(userId): void
ApplicationController
+getApplicationList(userId): List<Application>
+getApplication(applicationId): Application
+applyForJob(application, content): void
+editApplication(applicationId): void
+deleteApplication(applicationId): void
ApplicationRepository
+getApplicationList(userId): List<Application>
+getApplication(applicationId): Application
+applyForJob(application, content): void
+editApplication(applicationId): void
+deleteApplication(applicationId): void
PostRepository
+getPost(name): List<Post>
+getPost(workConditions): List<Post>
+postJobOpening(conditions, userId, email, childrenInfo, title, content): void
+postJobWanted(conditions, application): void
+getMyPostRecords(postId)
+editPost(postId) : void
+deletePost(postId): void
+postQnA(userId, title, qestionContent): void
+getAppliedPostRecord(userId): List<String>
Review
- Id: String
- rating: Int
- reviewRecipientId: String
- reviewerId: String
- reviewContent: String
- usedService: boolean
Application
- Id: String
- email: String
- title: String
- selfIntroduction: String
- attatchedFiles: List<String>
SitterPost
- applicationId: String
QnA
- Id: Stlng
- title: String
- questionContent: String
- answer: String
ParentPost
- applicationList: List<String>
PostController
+getPost(name): List<Post>
+getPost(workConditions): List<Post>
+postJobOpening(conditions, userId, email, childrenInfo, title, content): void
+postJobWanted(conditions, application): void
+getMyPostRecords(postId)
+editPost(postId) : void
+deletePost(postId): void
+postQnA(userId, title, qestionContent): void
+getAppliedPostRecord(userId): List<String>
UserRepository
+logIn(email, password): User
+signUp(userId, password, name, email, contactNum, childrenInfo): void
+signUp(userId, password, name, email, contactNum, gender, dateOfBirth, agreeToCctv, agreeToCriminalHistory): void
+forgotPassword(name, email, contactNum): void
SitterUser
- gender: Short
- dateOfBirth: Int
- agreeToCctv: boolean
- agreeToCriminalHistoryCheck:boolean
- applyHistory: List<String>
{User}
- Id: String
- name: String
- email: String
- contactNum : String
- favPostId: List<String>
Message
- date: date
- Id: String
- content: String
- hasBeenRead: boolean
ParentUser
- childrenInfo: String
{Post}
- Id: String
- date: date
- writerId: String
- title: String
- postContent: String
- workCondtion: WorkCondition
WorkCondition
- kindOfService:Stirng
- workPeriod: String
- workTime: String
- location: String
- gender: Short
- payRate: Int
UserController
+logIn(email, password): User
+signUp(userId, password, name, email, contactNum, childrenInfo): void
+signUp(userId, password, name, email, contactNum, gender, dateOfBirth, agreeToCctv, agreeToCriminalHistory): void
+forgotPassword(name, email, contactNum): void
MyPageController
+editProfile(name, email, contactNum, dateOfBirth, gender, agreeToCctv, agreeToCriminalHistoryCheck): void
+addToFavorite(userId, postId): void
+deleteFromFavorite(userId, postId): void
MyPageRepository
+editProfile(name, email, contactNum, dateOfBirth, gender, agreeToCctv, agreeToCriminalHistoryCheck): void
+addToFavorite(userId, postId): void
+deleteFromFavorite(userId, postId): void
MessageController
+sendMessage(senderId, receiverId, content): void
+markAsRead(messageId): void
+deleteMessage(messageId): void
+getMessage(messageId): List<Message>
MessageRepository
+sendMessage(senderId, receiverId, content): void
+markAsRead(messageId): void
+deleteMessage(messageId): void
+getMessage(messageId): List<Message>
1
0..*
1
0..*
1
1
0..*
1
1
0..*
1
0..*
1
0..*
1
0..*
1
0..*
1
1

Figure 2. UML class diagram

3.3 UML Sequence Diagram

https://app.lucidchart.com/invitations/accept/f2bb3c25-659a-46c9-be69-50ed7b400786

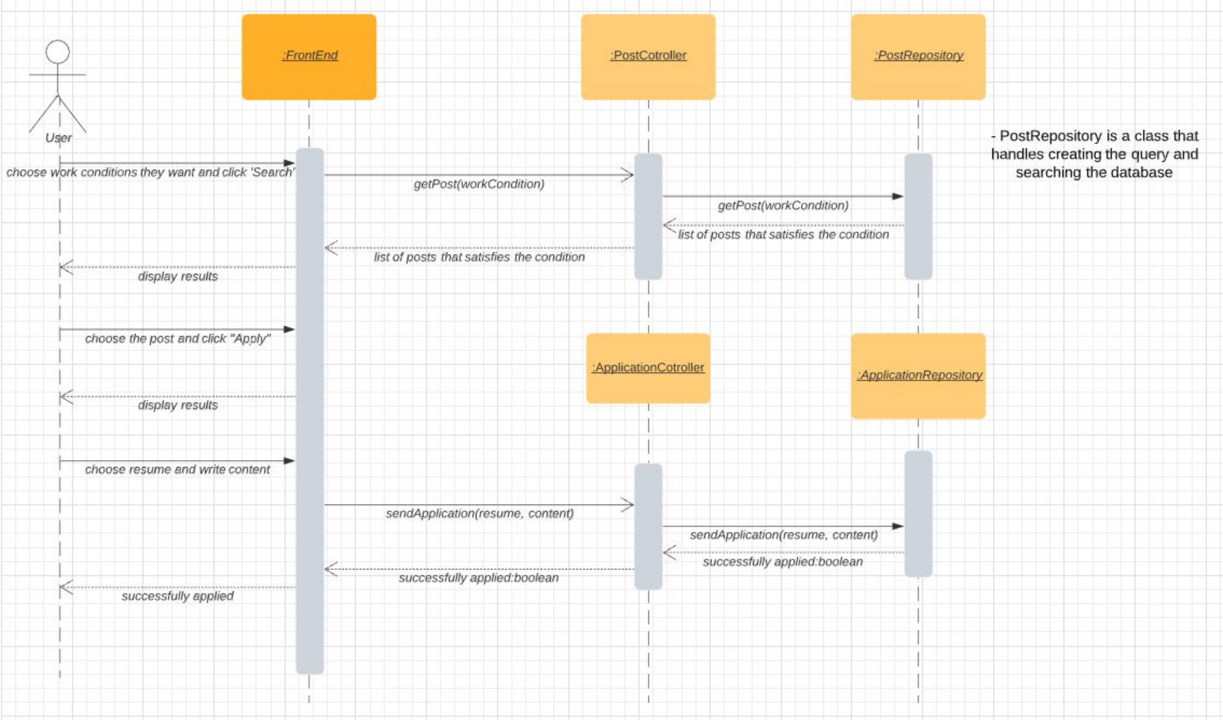
1. Parent reads an unread message and replies to the sitter.



- MessageRepository is a class that handles creating the query and searching the database

Figure 3. UML sequence diagram - Send Message

2. A sitter user finds a post and applies to the job.



:FrontEnd
:PostCotroller
:PostRepository
User
choose work conditions they want and click 'Search'
getPost(workCondition)
getPost(workCondition)
list of posts that satisfies the condition
list of posts that satisfies the condition
display results
choose the post and click "Apply"
:ApplicationCotroller
:ApplicationRepository
display results
choose resume and write content
sendApplication(resume, content)
sendApplication(resume, content)
successfully applied:boolean
successfully applied:boolean
successfully applied

- PostRepository is a class that handles creating the query and searching the database

Figure 4. UML sequence diagram - Search and Apply

3.4 Deployment

Our project has to be a dynamic website to reflect the changes made by our users in real-time. To do that, we decided to deploy our project using Heroku with remote MySQL through a package provided by Heroku for deploying a project built with Django.

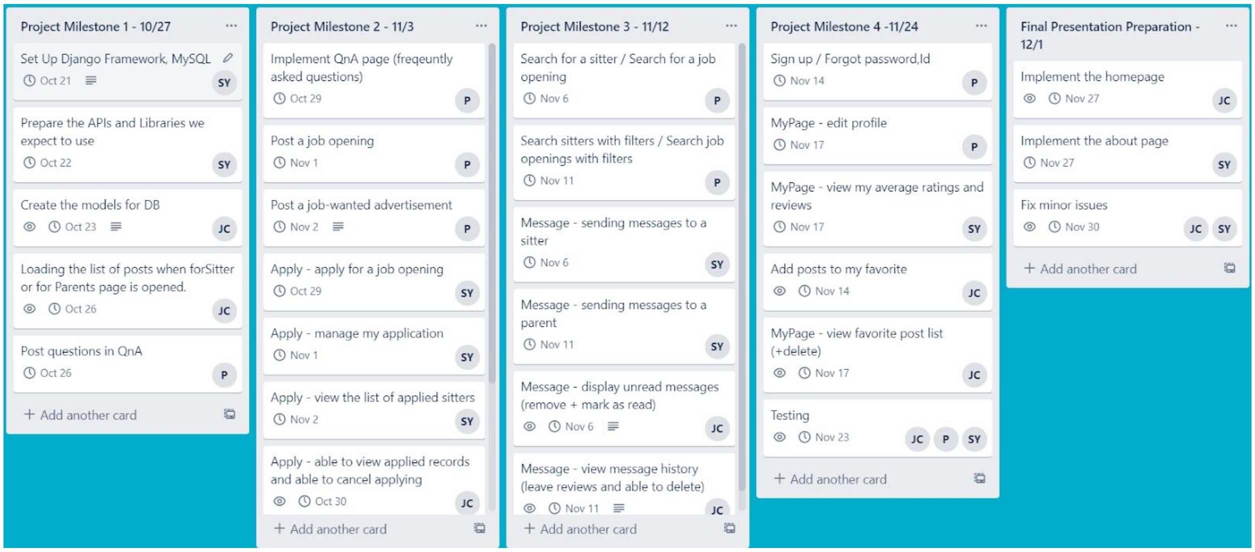
3.5 Code Conventions

JavaScript: Google JavaScript Style Guide
Python: PEP 8 -- Style Guide for Python Code.
Django: Django Coding Style
(https://docs.djangoproject.com/en/dev/internals/contributing/writing-code/coding-style/)

4. Schedule

These tasks are divided into a use-case level and some use-cases are divided into sub-tasks and our team tried to schedule important features first. We wrote a detailed schedule on the trello.

https://trello.com/invite/b/L58qOdgr/8bf375b1d0a7b9ac2b4064b010173e38/schedule-for-the-project



Project Milestone 1 - 10/27

- Set Up Django Framework, MySQL — Oct 21 — SY
- Prepare the APIs and Libraries we expect to use — Oct 22 — SY
- Create the models for DB — Oct 23 — JC
- Loading the list of posts when forSitter or for Parents page is opened. — Oct 26 — JC
- Post questions in QnA — Oct 26 — P

+ Add another card

Project Milestone 2 - 11/3

- Implement QnA page (frequently asked questions) — Oct 29 — P
- Post a job opening — Nov 1 — P
- Post a job-wanted advertisement — Nov 2 — P
- Apply - apply for a job opening — Oct 29 — SY
- Apply - manage my application — Nov 1 — SY
- Apply - view the list of applied sitters — Nov 2 — SY
- Apply - able to view applied records and able to cancel applying — Oct 30 — JC

+ Add another card

Project Milestone 3 - 11/12

- Search for a sitter / Search for a job opening — Nov 6 — P
- Search sitters with filters / Search job openings with filters — Nov 11 — P
- Message - sending messages to a sitter — Nov 6 — SY
- Message - sending messages to a parent — Nov 11 — SY
- Message - display unread messages (remove + mark as read) — Nov 6 — JC
- Message - view message history (leave reviews and able to delete) — Nov 11 — JC

+ Add another card

Project Milestone 4 -11/24

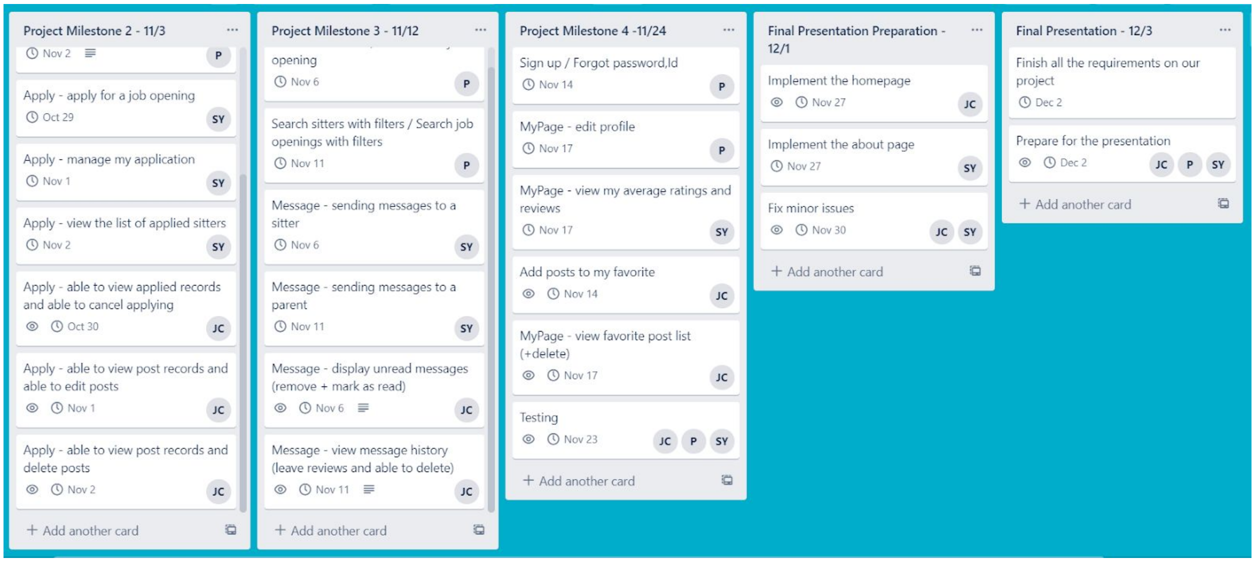
- Sign up / Forgot password,Id — Nov 14 — P
- MyPage - edit profile — Nov 17 — P
- MyPage - view my average ratings and reviews — Nov 17 — SY
- Add posts to my favorite — Nov 14 — JC
- MyPage - view favorite post list (+delete) — Nov 17 — JC
- Testing — Nov 23 — JC P SY

+ Add another card

Final Presentation Preparation - 12/1

- Implement the homepage — Nov 27 — JC
- Implement the about page — Nov 27 — SY
- Fix minor issues — Nov 30 — JC SY

+ Add another card



Project Milestone 2 - 11/3

- Nov 2 — P
- Apply - apply for a job opening — Oct 29 — SY
- Apply - manage my application — Nov 1 — SY
- Apply - view the list of applied sitters — Nov 2 — SY
- Apply - able to view applied records and able to cancel applying — Oct 30 — JC
- Apply - able to view post records and able to edit posts — Nov 1 — JC
- Apply - able to view post records and delete posts — Nov 2 — JC

+ Add another card

Project Milestone 3 - 11/12

- opening — Nov 6 — P
- Search sitters with filters / Search job openings with filters — Nov 11 — P
- Message - sending messages to a sitter — Nov 6 — SY
- Message - sending messages to a parent — Nov 11 — SY
- Message - display unread messages (remove + mark as read) — Nov 6 — JC
- Message - view message history (leave reviews and able to delete) — Nov 11 — JC

+ Add another card

Project Milestone 4 -11/24

- Sign up / Forgot password,Id — Nov 14 — P
- MyPage - edit profile — Nov 17 — P
- MyPage - view my average ratings and reviews — Nov 17 — SY
- Add posts to my favorite — Nov 14 — JC
- MyPage - view favorite post list (+delete) — Nov 17 — JC
- Testing — Nov 23 — JC P SY

+ Add another card

Final Presentation Preparation - 12/1

- Implement the homepage — Nov 27 — JC
- Implement the about page — Nov 27 — SY
- Fix minor issues — Nov 30 — JC SY

+ Add another card

Final Presentation - 12/3

- Finish all the requirements on our project — Dec 2
- Prepare for the presentation — Dec 2 — JC P SY

+ Add another card

Figure 5 & 6. Schedule

5. Contributions

Ji Won Choi

- Revised Requirements part (2.1, 2.3, 2.4)
- Add 3.5 Code convention
- Design 3.2 UML Class Diagram
- Design 3.3 UML Sequence Diagram
- Scheduled the project
- Finishing revision of the document

Sujeong Youn

- Applied the comments from Professor for Introduction (1.1 ~ 1.7)
- 3.1 Overview research and writing
- 3.4 Deployment research and writing
- Participated in scheduling of project
- Design 3.2 UML Class Diagram
- Design 3.3 UML Sequence Diagram
- Finishing revision of the document

Pyungkang Hong

- Design 3.2 UML Class Diagram
- Design 3.3 UML Sequence Diagram
- Revised 2.2 Use Case Diagram