DELIVERABLE 4

SiteSync

SiteSync Site Management

Team Number 13

- Anthony Mancia (N01643670)
- Chris Garcia(N01371506)
- Ngoc Le (N01643011)
- Tyler Meira (N01432291)

Name	Student ID	Github ID	Signature	Effort
Tyler Meira	N01432291	95157172		100 %
Chris Garcia	N01371506	196109735		100 %
Ngoc Le	N01643011	157065810		100 %
Anthony Mancia	N01643670	195244864		100 %

9. Brief description of the project, couple sentences.

The siteSync Management project is a project to provide construction workers a platform in which they can find, manage, and track their jobs around the GTA. It aims to eliminate the hassle of looking for jobs aimlessly, and simplifies tracking hours worked for employers to employee transparency.

10. What issue your product will solve.

Our product solves the issue of construction workers struggling to find jobs in their respective fields, and the lack of employer accountability when it comes to tracking employee hours. Our product solves both those issues in regards to the construction industry.

11. Compare your application with at least two existing apps in the market. Provide links and description of the two apps you selected.

Two applications that our product competes with are Homebase and Clockify, both these softwares provide functionality similar to our application that allows users to clock in and out the logs the times.

http://joinhomebase.com/
https://clockify.me/

12. Highlight the differences between your app and these two apps.

The difference between our app and our competitors app is the we also, have a hardware component that would come as part of our package, our app has intended functionality not let the users clock in from their phone and interact with the hardware component of our app.

13. Why you believe your app is better than these two apps.

We believe that our app is better that these two apps because, our apps forces employees to be at the job site, this helps ensure accountably for the employee and peace of mind for the employer without them having to physically be there.

14. Create a table pros and cons of the three different apps.

System	Pros (Strengths)	Cons (Weaknesses)	Primary Focus Missing in CTG
SiteSync	Mandatory On-Site Accountability via proprietary hardware component. Solves two problems (job finding + time tracking) in one vertical platform.	Initial investment required for hardware. Focused on job to job based contracts.	Broader, general-purpose HR/Payroll management.

QuickBooks	Excellent GPS tracking and mileage logging. Deep integration with payroll and accounting software (QuickBooks).	No compulsory hardware check. Users can manually edit clock-in location. High monthly subscription cost.	Specific job matching/finding features for skilled trades.
Homebase	Strong free tier for small businesses. Great tools for scheduling and team communication. Excellent compliance checks for breaks/overtime.	Focuses on retail/restaurant/small service businesses. No hardware check; relies heavily on soft location data.	Lacks industry-specific features like trade-based job finding.
Clockify	Excellent free option. Simple to use for project-based time tracking. Good reporting features for non-hourly work.	Designed primarily for freelancers and office/remote workers. Lacks strong payroll/HR integration and physical accountability features.	Payroll integration, physical accountability, and job searching.

15. GitHub Repo link. All members must contribute to the repo.

https://github.com/ChristopherGarcia1506/SiteSync.git

16. Verify the link is working.

17. Login functionality, I will use the following credentials to test your app: Email: aaa@bbb.com Password: Admin101!
Admin:
Email: aaa@bbb.com Password: Admin101!
Employee:

Email: lionelmessi@hotmail.com: Password:Admin101!

18. Document any other login credentials I must use i.e. Admin vs. regular user. I should not need to send you an email to login to your app.
19. You are working on sprint 4. CENG-322 2

20. Describe in detail, the work that has been completed by each team member in this sprint only.

Chris Garcia:

For sprint 4, I added further functionality to the Register Screen. When a user registers, their account is also added to the firestore authentication page. Furthermore, I worked on redesigning and implementing aspects of the Profile Screen, such as Edit Profile, and Logout buttons. I also worked on displaying the jobs posted by the employer to their home screen, for them to keep a log of the jobs they have posted. The Profile Screen also now displays the users information, such as their names, email, phone number, and organization.

Tyler Meira:

For this sprint I continued the implementation of the Job board, Active jobs and Past Jobs fragments, I also implemented a new Fragment Punch Log Fragment. In the Job board fragment I added a onclick listener to that will display a dialog box asking if the current logged in user would like to accept one of the active jobs, if they click yes they get added to job employees array in the database, then the active and past jobs will display all the current employees jobs that they have signed up for, and completed jobs once they are done / InActive, I create a new collection in the database to simulate the data from our hardware project, the hardware project will store this data in the database, then we will pull that data and display in our app, the new fragment Punch Log Fragment, will display the current logged in users punches and if the user is a employer it will display the punches for all the employees that are working their jobs.

Ngoc Le:

I added validation for password in the register screen such as making sure the password has a minimum length of 6 characters, the password now must have at least one uppercase letter, one digit and one special character. I also added validations for email and phone number for the feedback screen, and also made sure all fields are filled. The feedback screen also has a progress bar and alert dialog to notify users if the submission is completed. I also added a password change functionality for the user's profile screen.

Anthony Mancia: Reworked the Permissions tab to have better functionality by adding the option to disable snackbars on top of disabling the exit dialog permission upon every time you wanted to leave the app, Additionally, I added the 10 java test cases to test the app for example testing the overflow menu at the top right and by clicking on it and opening permission and also opening the FAQ.

21. Each member must have a minimum of 10 commits, counted starting Nov. 3 (Two-week Sprint).

Chris Garcia: 10

Tyler Meira: 15

Gnoc Le: 10

Anthony Mancia: 10

22. Marks deducted for members who are not meeting the minimum 10 commits (i.e. if only 8 commits, 20% percent deducted, …etc.). The work should be done over the sprint, and not in the last 48 hours prior to the submission.
23. List number of commits for each member for this sprint only. Create a table like below: Team Member Harpreet Eric Arya Nov 4 4 2 0 …. … … … Total 15 12 14

Team Members	Chris Garcia	Tyler Meira	Ngoc Le	Anthony Mancia
Nov 16th	4	5	10	
Nov 15th	5	7	4	
Nov 14th		1		
Nov 13th		1		
Nov 12th	1	1		
Total		15	14	

24. Sprint goals, list sprint goals.

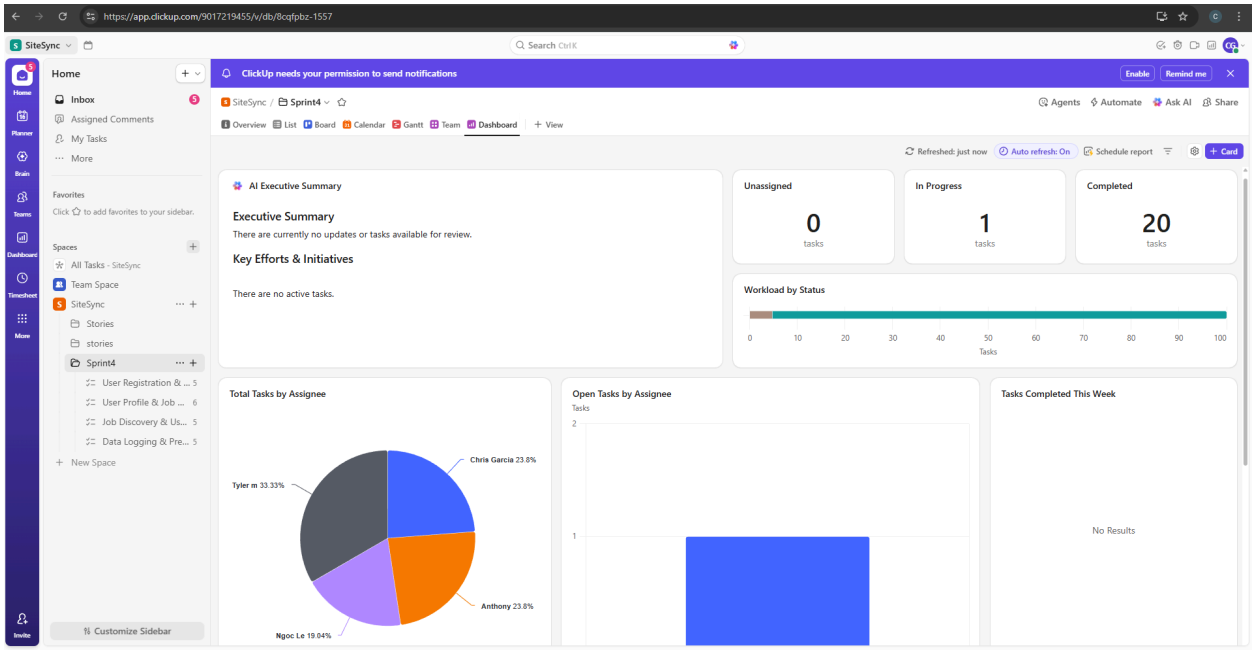
1. Display the corresponding jobs to employers
2. Create Time Log Screen and collection
3. Complete Profile Screen UI
4. Complete Profile Screen Functionality
5. Implement Full Job Board Functionality
6. Implement Active and Past Job display for employees
7. Add punch log collection
8. Implement activity to show users punches.

25. Update the Sprint dashboard, showing Sprint 4 with closed tasks and tasks you did not complete.

26. Dashboard, should clearly show task, owner, status, start date, end date, size and priority.

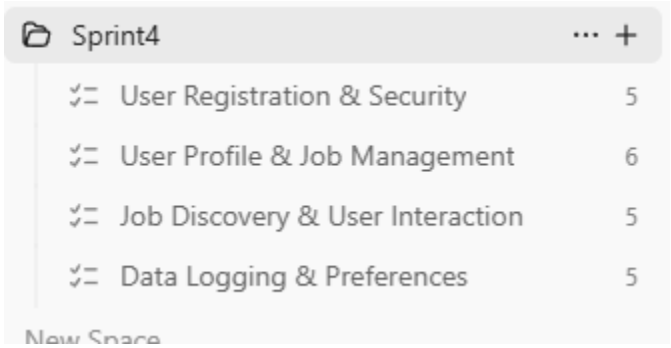
27. Must use a tool such as Trello or Monday. Word, Excel, … etc. will not be accepted.

28. Please use Scrum and not Kanban. An example of Scrum board, add column priority!



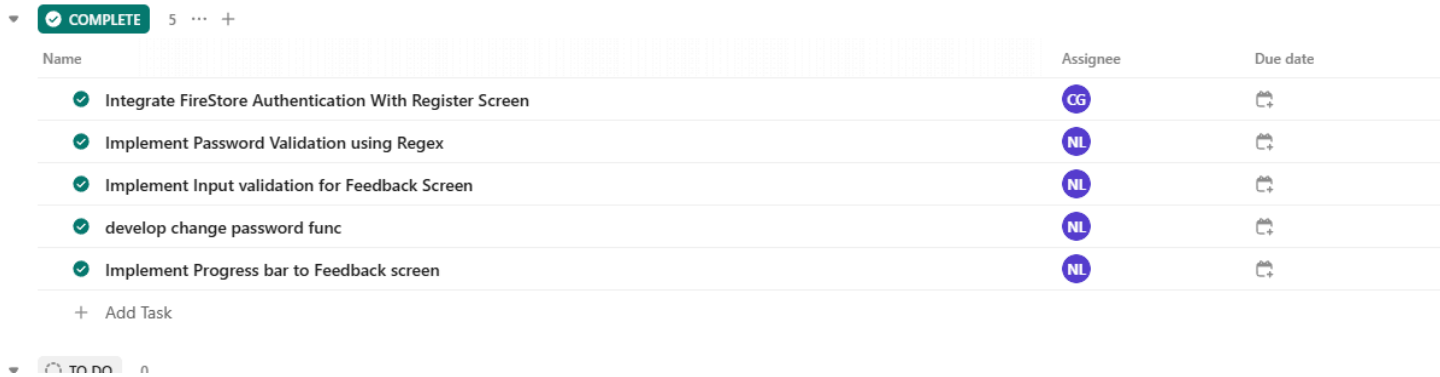
29. Take a screenshot showing clearly the stories and breakdown of tasks with all details for sprint 4 only.

Stories:

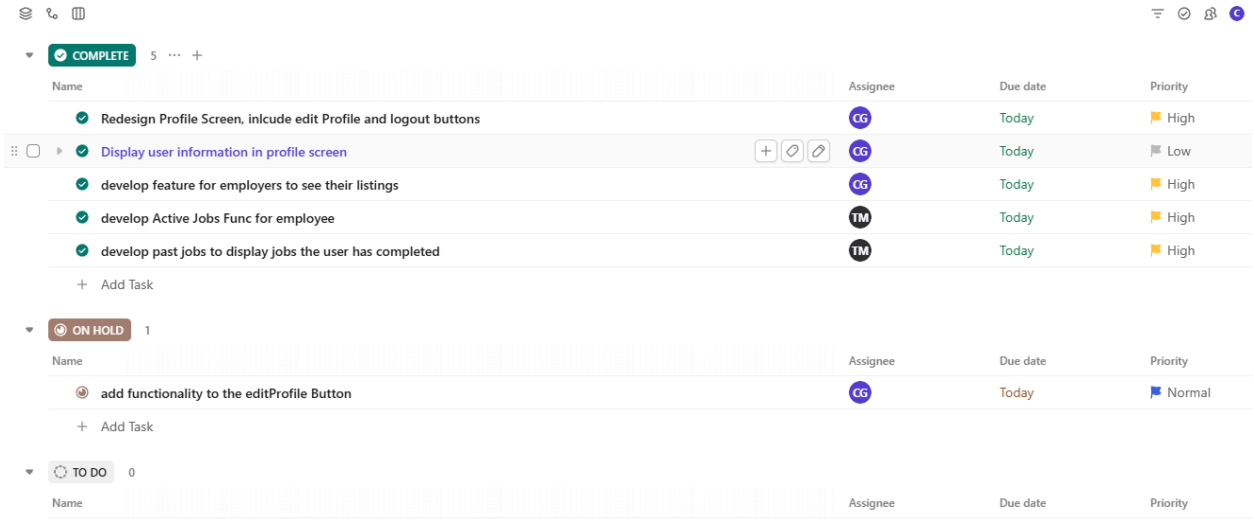


30. Must have a minimum of 4 stories and 5 tasks per story.

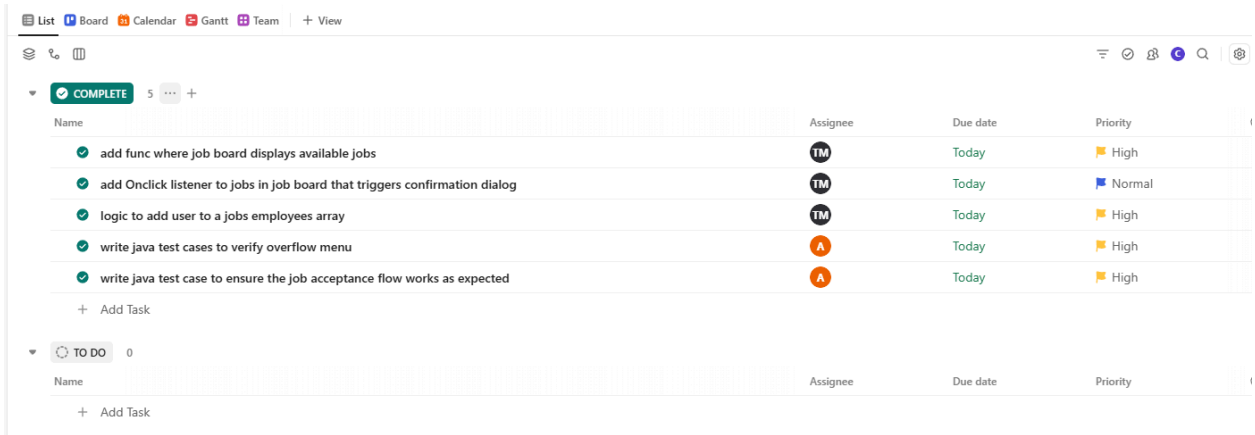
User Registration & Security



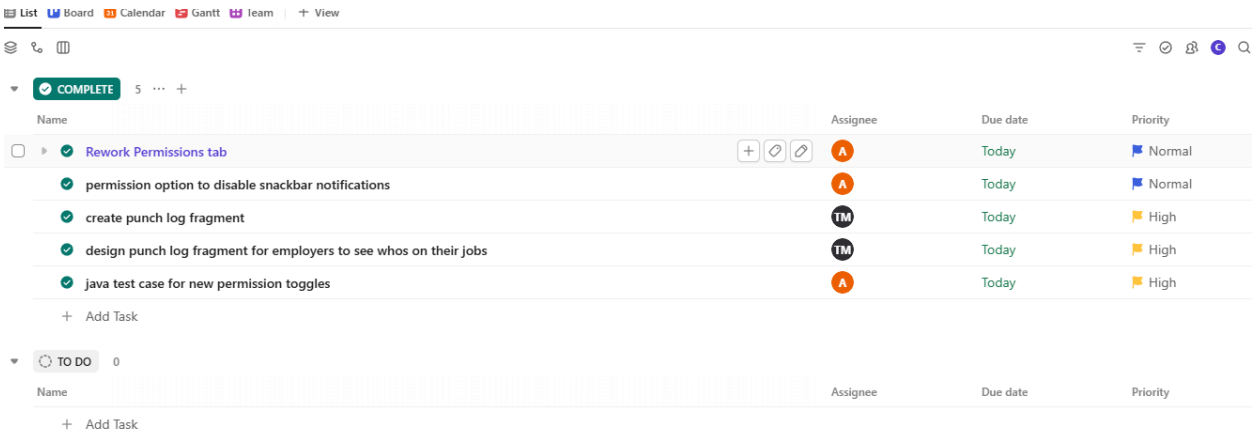
User profile & Job Management



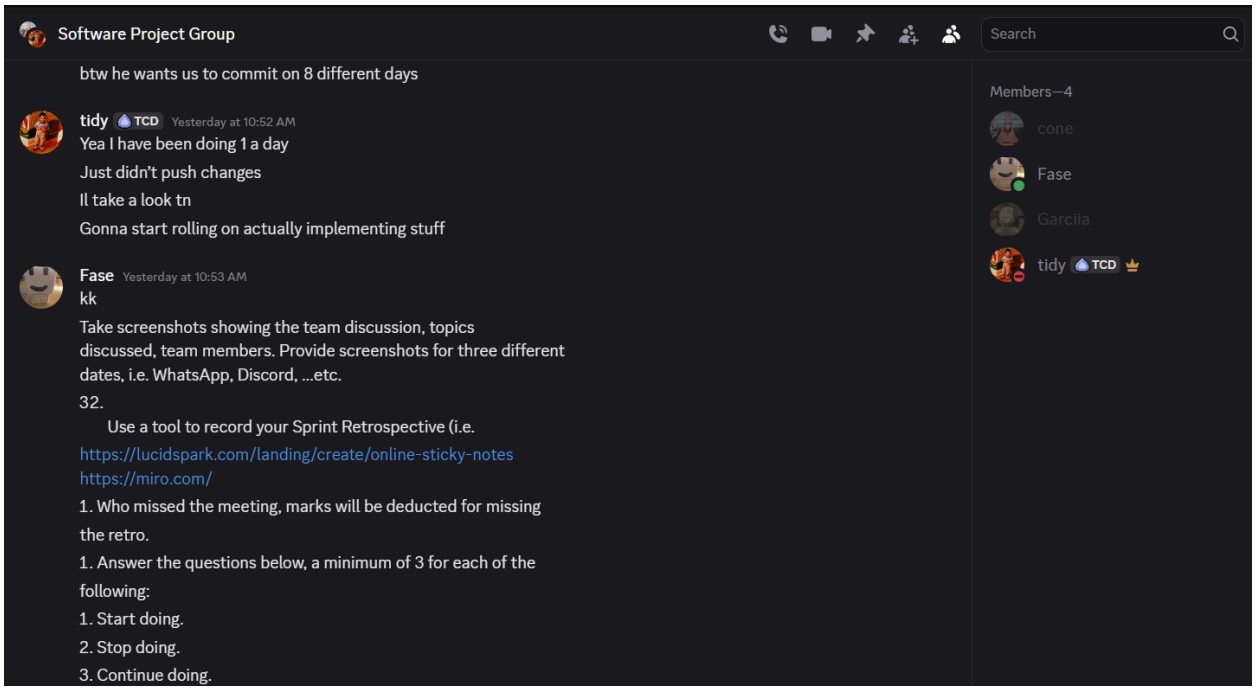
Job Discovery &USer Interaction



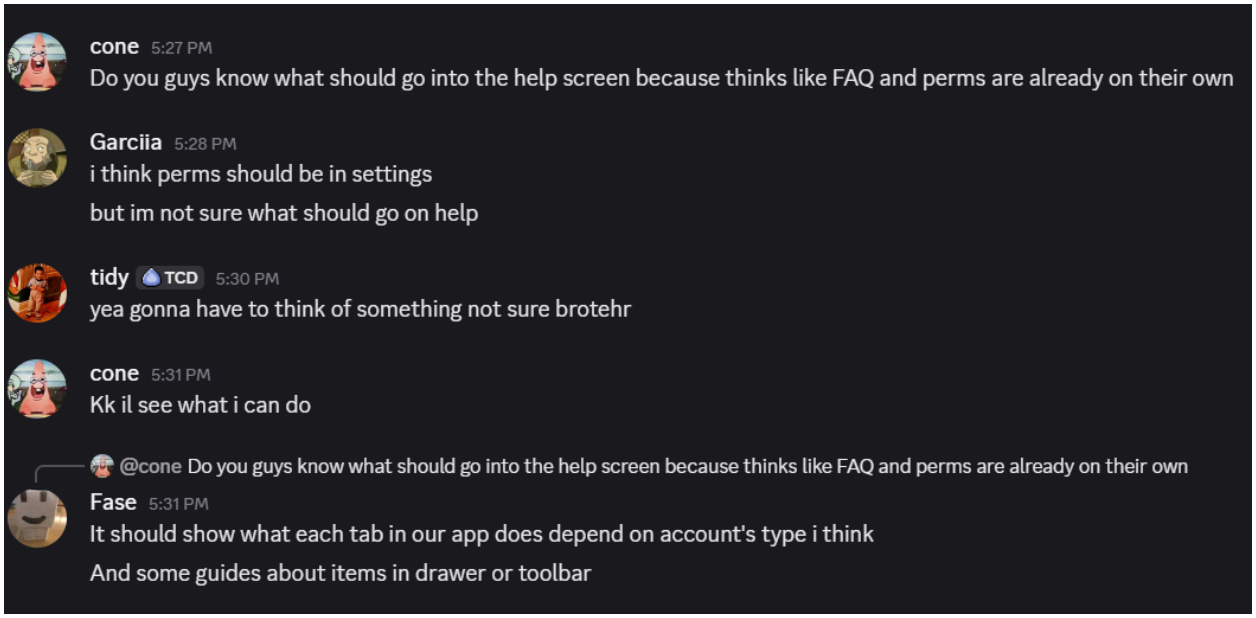
Data Loggin & preferences



31. Take screenshots showing the team discussion, topics discussed, team members. Provide screenshots for three different dates, i.e. WhatsApp, Discord, …etc.

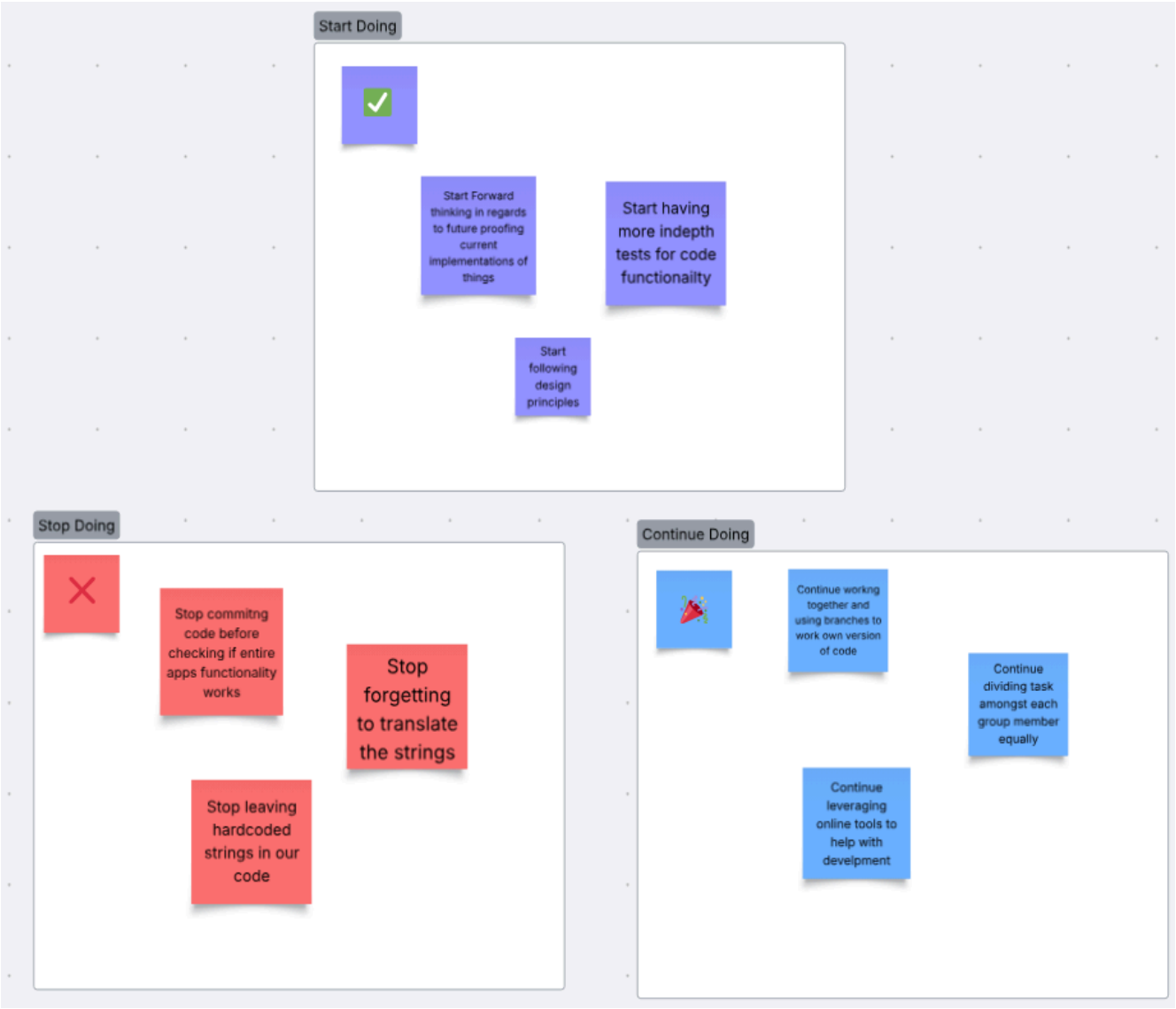


We talked about the requirements of the deliverable on November 16th.

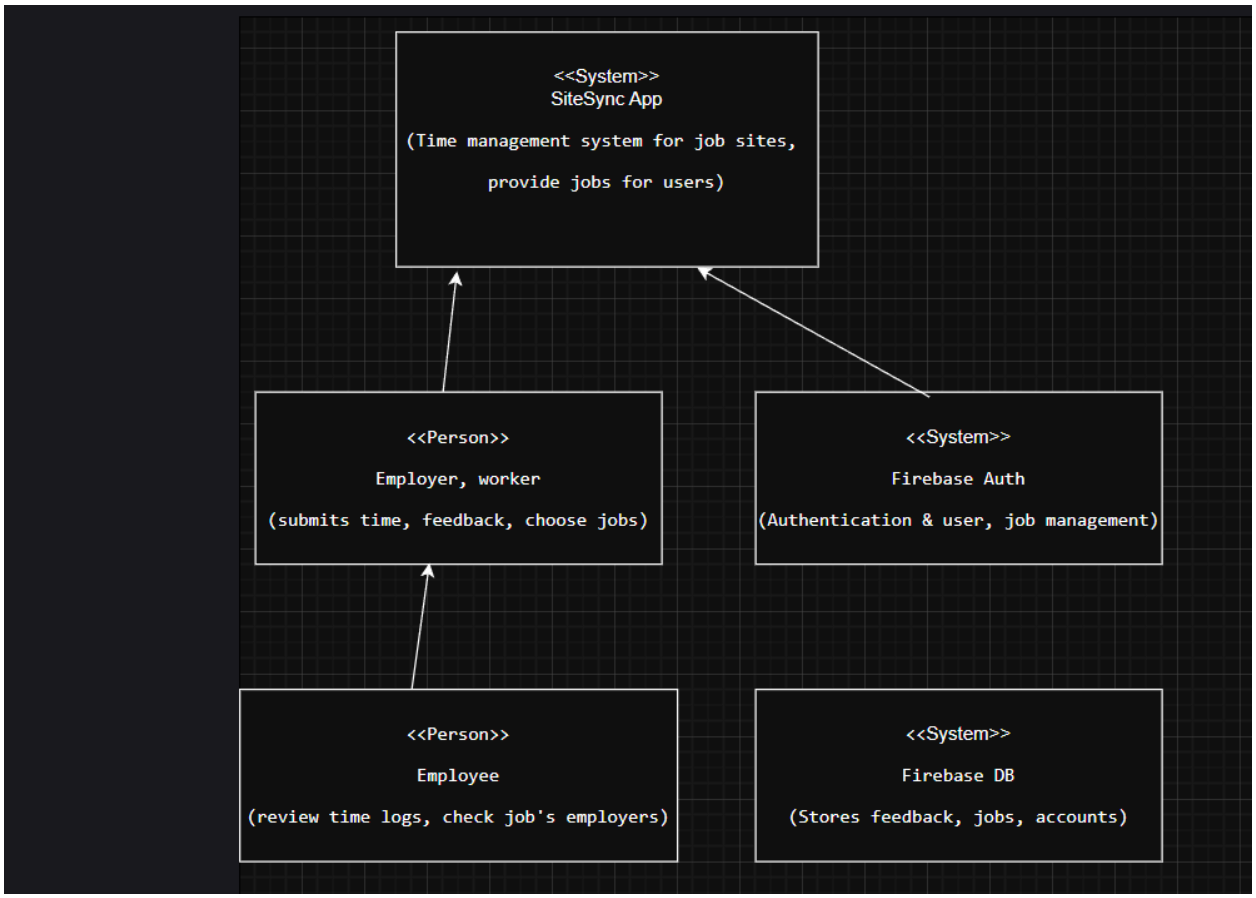


We discussed the help fragment in our app such as what should go into the help screen on November 17th.

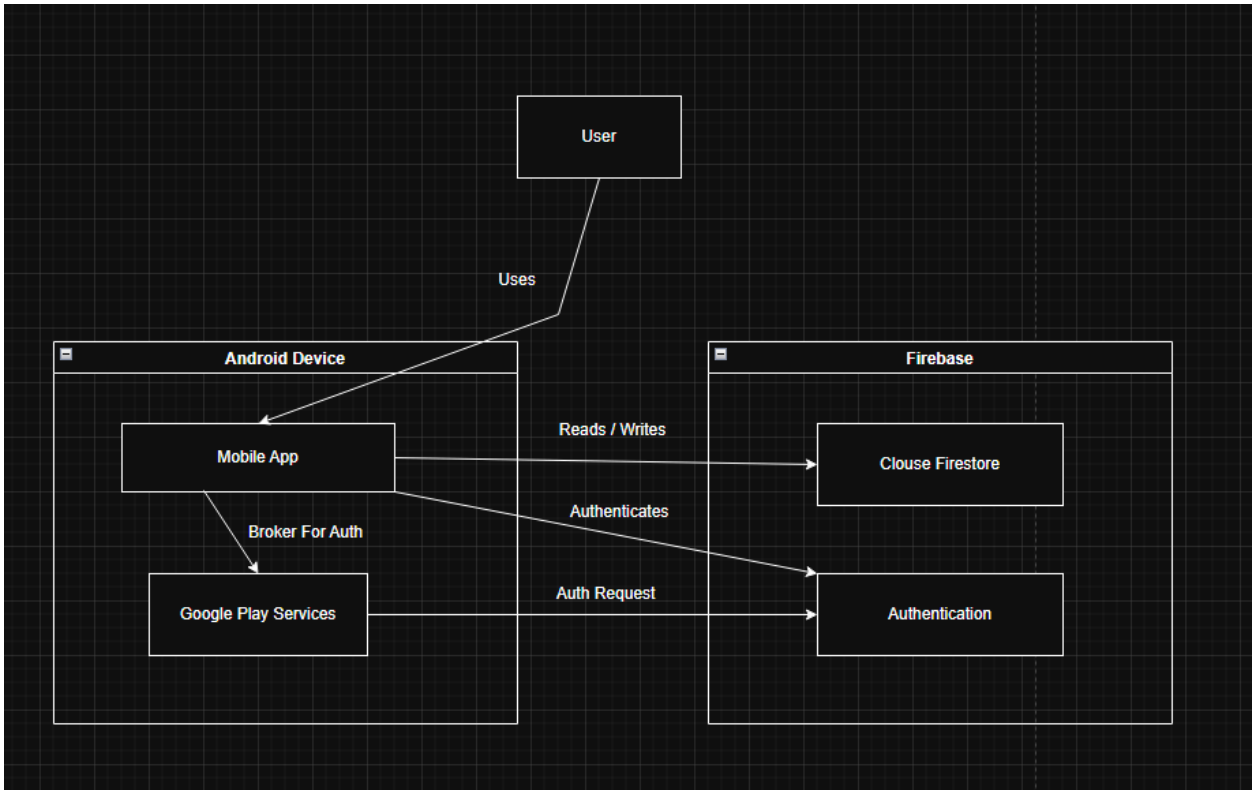
32. Use a tool to record your Sprint Retrospective (i.e. https://lucidspark.com/landing/create/online-sticky-notes https://miro.com/ 1. Who missed the meeting, marks will be deducted for missing the retro. 1. Answer the questions below, a minimum of 3 for each of the following: 1. Start doing. 2. Stop doing. 3. Continue doing. 2. Take a screenshot. CENG-322 4 3. Must use online tool for the Retrospective. No mark will be given if no tool used.



33. Using C4 Model, show "System Context Diagram". Use a tool to draw your diagram, hand drawing will not be accepted, i.e. you can use https://app.diagrams.net/



34. Using C4 Model, show "Container Diagram". Use a tool to draw your diagram, hand drawing will not be accepted, i.e. you can use https://app.diagrams.net/



35. Document two different design patterns used in the code. Copy the code you used, and add your explanation. Use the ones covered in the class.

We use the adapter pattern, something that we learned in the mobile programming is the use of an adapter class to fill recycler views

```
public class JobAdapter extends RecyclerView.Adapter<JobAdapter.JobViewHolder> {

    private List<JobItems> jobList;

    public JobAdapter(List<JobItems> jobList) {
        this.jobList = jobList;
    }

    @NonNull
    @Override
    public JobViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.items_job, parent, false);
        return new JobViewHolder(view);
    }

    @Override
```

```
    public void onBindViewHolder(@NonNull JobViewHolder holder, int position) {
        // Get the data object from your list.
        JobItems job = jobList.get(position);

        // Translate the data into UI elements.
        holder.companyName.setText(job.getCompany());
        holder.description.setText(job.getDescription());
        holder.status.setText(job.getStatus());
    }

    @Override
    public int getItemCount() {
        return jobList.size();
    }
```

The second pattern is the Observer Pattern, this pattern defines a one-to-many dependency between objects so that when one object changes state, all its dependent are notified and updated automatically.

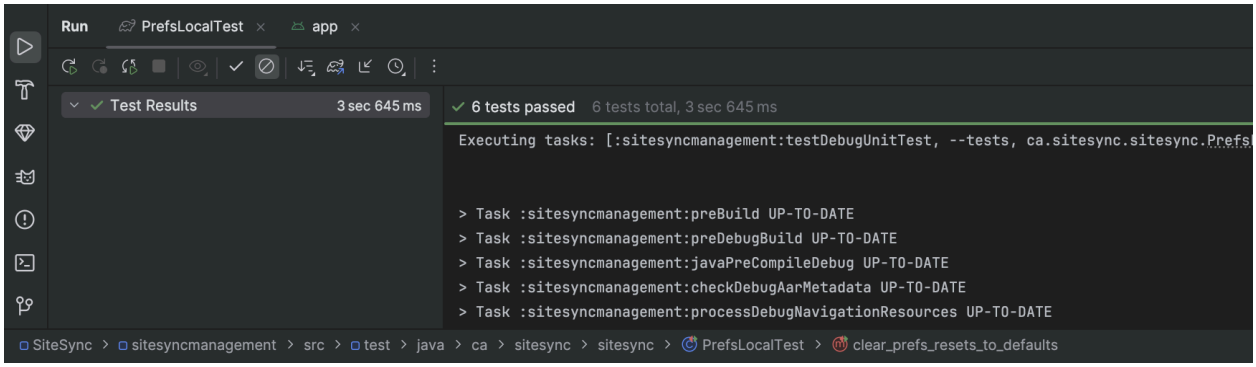
```
public class JobAdapter extends RecyclerView.Adapter<JobAdapter.JobViewHolder> {

    private List<JobItems> jobList;

    public interface OnItemClickListener {
        void onItemClick(JobItems jobItem, int position);
    }

    private OnItemClickListener listener;

    public void setOnItemClickListener(OnItemClickListener listener) {
        this.listener = listener;
    }

    @Override
    public void onBindViewHolder(@NonNull JobViewHolder holder, int position) {
        JobItems job = jobList.get(position);
        holder.companyName.setText(job.getCompany());

        holder.itemView.setOnClickListener(v -> {
            if (listener != null) {
                listener.onItemClick(job, position);
            }
        });
    }
```

36. Your code should take Design Principles and Design Patterns into consideration, the ones covered in the class.

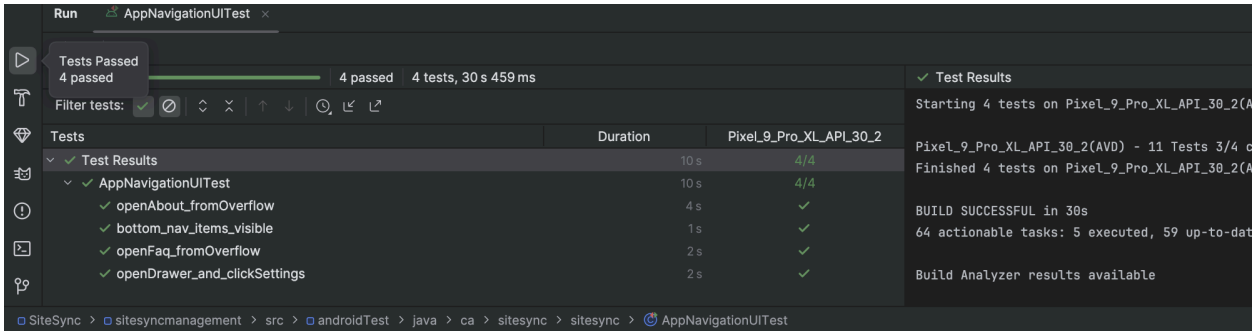
37. Coding work progress since deliverable 3. What additional features/functionality added since deliverable 3.

38. Write unit test cases. Select one of your Java classes and write a minimum of 10 test cases.

JVM(Local Test): 6 Tests



UI Test: 4 Test



39. Demonstrate the use of assertEqual, assertTrue, assertFalse, and asssertNotNull in your testing.
40. All test cases must pass.
41. Take screenshot showing all you test cases are passing.
42. Functionality on the Customer Feedback Screen below.
43. Display an error and don't submit if invalid input, i.e. invalid phone number, …etc.
44. Display progress bar while the info is getting submitted, implement some delay for 5 seconds.
45. Once you receive a confirmation from the DB, display an AlertDialog with OK confirming the form has been submitted.
46. Verify you are submitting device model and stored into DB. You extract device model programmatically (don't ask for device model in the form). CENG-322 5
https://www.tutorialspoint.com/how-to-check-android-phonemodel-programmatically
47. Add into pdf file screenshot showing the progress bar while the form is gekng submiled.

48. Add into pdf file screenshot showing the AlertDialog once the form is submiled successfully. 49. Add into pdf file screenshot showing the data stored into the DB. Must have at least 3 different entries.
50. Clear the user input once the form is submitted. Restrict user submission to once per 24 hrs. 51. Once the user submits the form successfully, gray out the submit button, and display a timer showing how many hours and minutes remaining when the user can submit another feedback. 52. Describe in the pdf file on how you satisfied this requirement.
53. Take screenshot showing all sensor data fetched and updated from the DB.
54. Must implement at least two features of your application, they are related to the core functionality. Describe what main functionality added.
55. Describe the main functionality added in this sprint.
56. Create a subfolder called deliverable4 under docs in the repo and add the pdf file