CEN4725/5726 Natural User Interaction

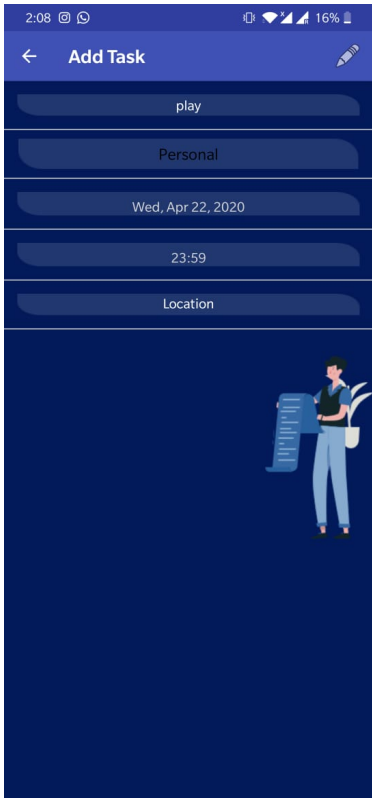
Course Project Final Paper

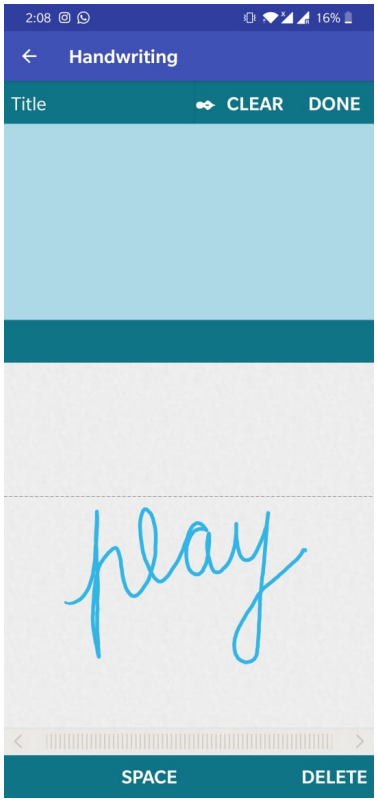
Group Members:
[Arushi Rastogi] [Rujuta Hajarnis]
[Apoorva Agarwal] [Bharath Shankar]

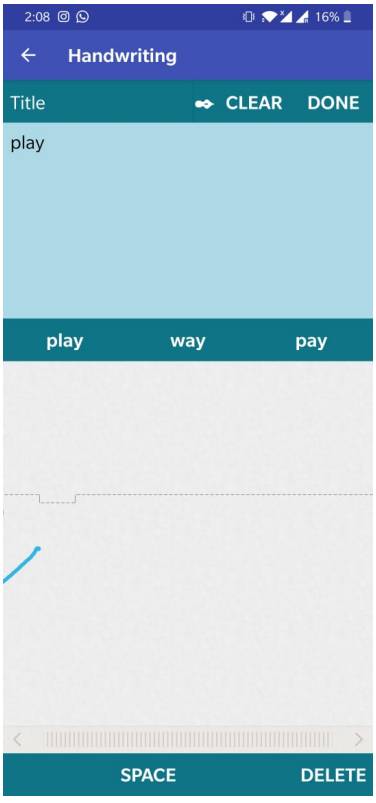
Section: Graduate

To-Do List App with Gestures + Handwriting Recognition

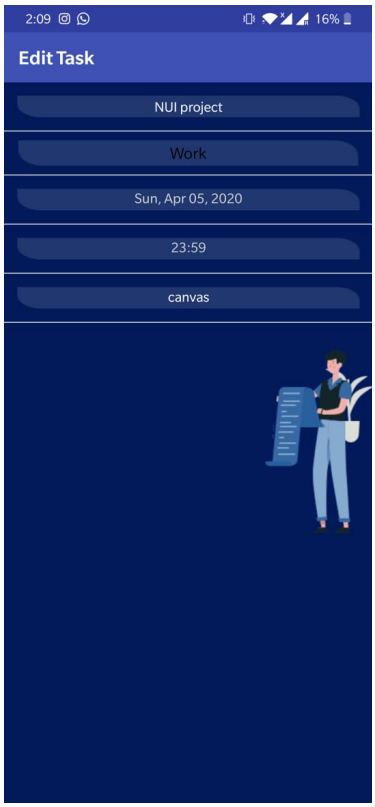
Create task: Handwriting recognition(1): Handwriting recognition(2):

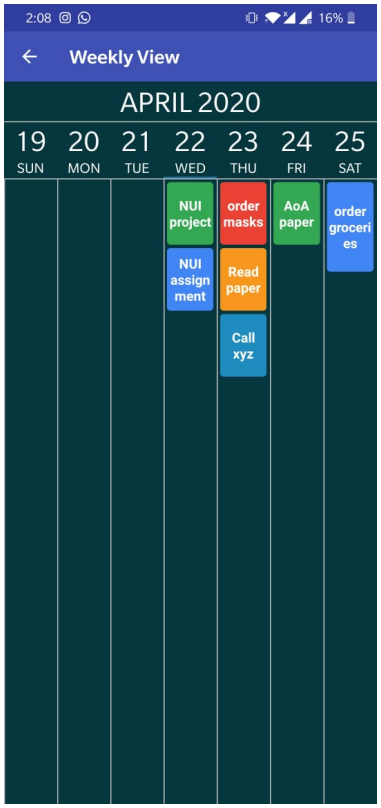


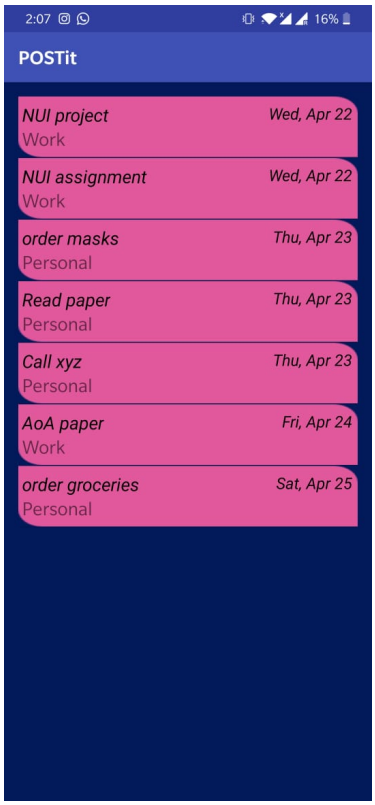




Edit task: Weekly view: List view:







1 Iterative Design and Development Process

1.1 Initial Design

The initial design involved a lot of brainstorming about the use of the app amongst team members. Since the to-do list app can be used by people of different age groups and professions, various personas who would use the app were considered. It included people from various age groups. The **personas** considered also had varying interaction and skill with technology which allowed us to design the app with diverse considerations. The major ones were:

1. **Middle-aged working person:** Moderately versed with technology, would sometimes need help with learning new technology/apps, handwriting or typing would be based on personal preference.

2. **Old-aged person:** Very less interaction with technology, will need to be taught to use new apps, natural gestures would be most useful, handwriting would often be preferred over typing.
3. **Student/youth:** Well versed with technology, could easily learn and use new apps, gestures might be helpful for quicker use.

Since we were developing an app, we decided to go with on-screen gestures that could be drawn on the touch screen. These were the most natural and didn't require any additional hardware. They could also be drawn at any place within the range of the screen, thereby allowing easy transition.

Interviews:

1. Raj (Middle-aged working person) - Telephone interview.

Q) How often do you use a reminder or to-do list app?
Ans. Not too often but maybe sometimes for groceries.

Q) Would you prefer all the basic commands to be performed using simple gestures or buttons?
Ans. I guess gestures would certainly help because sometimes we aren't sure where the different options in an app are to be found with various menus and buttons.

Q) What about handwriting recognition, where you can directly write on to the screen of the app?
Ans. I believe that would make things more convenient because many of us prefer writing in a notepad and this would make things very similar. Besides we are much slower in typing on the smartphone than the younger generation.

Q) What gestures would you prefer for the app?
Ans. I'm not too sure but anything that is easy to draw and remember should work just fine.

Q) What fundamental functions do you think the app should include?
Ans. I should be able to add items to a list, delete it, view details of the list and tasks.

2. Suresh(Old-aged person) - Telephone interview.

Q) How often do you use a reminder or to-do list app?
Ans. I haven't actually used it till now.

Q) Would you prefer all the basic commands to be performed using simple gestures or buttons? (After explaining how gestures work on smartphones)
Ans. I think gestures will be much easier since it's just simpler but it might take some time getting used to.

Q) What about handwriting recognition, where you can directly write on to the screen of the app?
Ans. Yes, I think I'll very much prefer that since I have always preferred writing with a pen and paper over smartphones or laptops. It is much quicker and easier

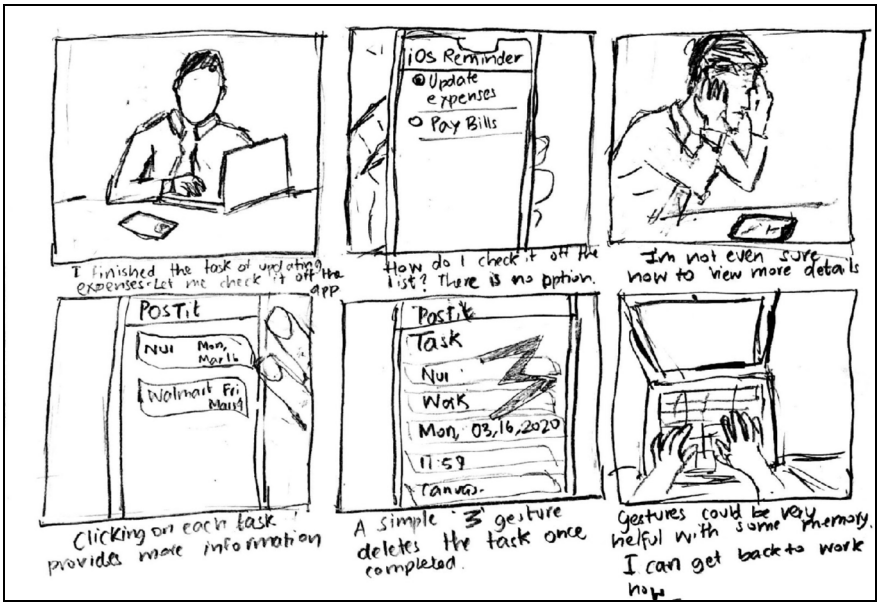
Q) What gestures would you prefer for the app?
Ans. I don't really know which ones would be good. But most importantly it must be easy to draw and remember.

Q) What fundamental functions do you think the app should include?
Ans. All the basic things I can do in a physical book to help me remember (laughs).

Storyboard:

Old-aged person

The storyboard shows that an old-aged person finds gestures easier to navigate an app as compared to various menus.



Middle-aged working person

The storyboard shows that some apps do not provide obvious functionality explicitly and how gestures could solve the problem. (NOTE: Delete Task was changed to Completed Task with '✓' gesture)

For the initial prototype design, we implemented 4 of the most basic gestures needed to perform essential functions of the app. These gestures were based on what users could easily draw and at the same time, matched with the task itself.

- For the new task option, we used the gesture of the character "∩". It is similar to a small n. It allowed users to quickly enter details for a new task that is to be done with just the "∩" drawn at any screen of the app. ∩ was chosen because it would be easier for users to remember.
- To save the task, we used the "✓" since it was intuitive with the gesture that is generally used for saving.
- For the delete function, we believed the inverted sigma gesture allowed a quick way for users to cancel out a particular task. Also, it symbolizes the way we erase something, so it is easier for people to remember. When marking something to be wrong, this gesture is normally used and would aid with memory and association.
- Finally, to go to the previous page we have used a "horizontal curve (drawn from the right)" which is again related to the task performed.

With these functions being implemented, a basic version of the app could be run allowing creation, deletion, and saving of tasks.

1.2 Developing the First Prototype

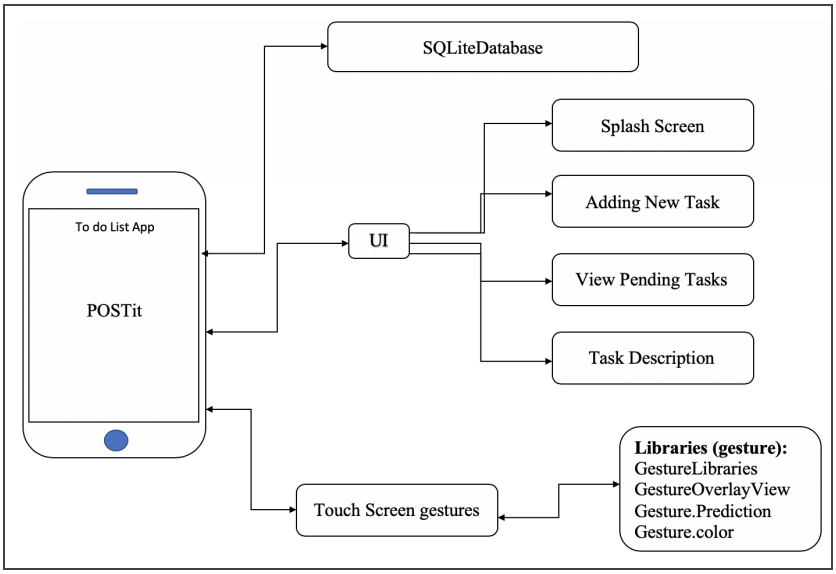
Based on the design developed by considering users of various age groups, the first prototype was developed. The development process included the following steps :

- We identified the basic functionalities that needed to be performed. These included creating a new task, saving the created task, and editing the already created tasks.
- After identifying the tasks to be performed, we discussed the possible gestures that we could use for the respective tasks. We also asked around for getting some insights from people of different demographics. This included friends and family.
- Once we had collected a good amount of data, we physically sketched out how our app would look like using Sketch. Bharath was in charge of the design process.
- Iteratively many changes were made to the design. After finalizing the design, we started with the app coding. This was mainly done by Rujuta, Apoorva, and Arushi.

- For incorporating the gestures, we decided to use PDollar recognition since we had studied it in the class and had an experience with its implementation in the first assignment.
- Once the app was fully functional, we started working on the documentation part. All four of us were equally involved in the documentation which included the report and presentation.

API's, Toolkits and frameworks used include: Sketch, Android Studio, Android SDK, Gradle, SQLite, Language used: Java

System Architecture Diagram:



1.3 Developing the Second Prototype

After developing the first prototype, we had a basic outline of how we wanted our app to look like. It was used as a skeleton to build the final version of the app. The feedback received from Dr. Ruiz and the other users aided to add features and improve the app after analyzing feasibility and value. A development process similar to the first one was used to implement these features. Suitable gestures had to be identified after which iterative changes to the code were made using GitHub for version control and PDollar for gesture recognition.

1.3.1 Change 1: Changes to the existing prototype

- Since the tasks in a to-do list are not deleted and are rather completed, we decided to remove the delete functionality gesture(‘inverted sigma’) and incorporate the complete functionality instead. This was an example of changing feedback of a NUI command.
- The gesture of completion was set to be a ‘✓’ symbol, which was previously used to save tasks. ‘✓’ depicts completion and was hence used.
- After some contemplation, we thought that a simple ‘S’ symbol would be a better choice of gesture for saving the task. It would also be easy on the user’s memory.

1.3.2 Change 2: New additions to the prototype

- Apart from the existing functionality, we added some more features to the to-do list app.
- These features include filtering between personal and work(using ‘p’ and ’w’) tasks, getting a weekly view(‘pinch’ and ‘zoom’ to go back), and being able to edit the already existing tasks with a ‘Σ’ symbol.
- The idea to include some of these were from user-feedback, we believed that filtering and displaying the tasks in different views would make the app more useful and efficient. And editing a task was a necessity for mistakes during task creation.
- Hence they were implemented using the above apt gestures that could be drawn on the screen and were inherent as they were the first letter of the words. ‘Σ’ was also similar to the ‘E’ for Edit. The pinching gesture seemed intuitive to get an overall view of the week.

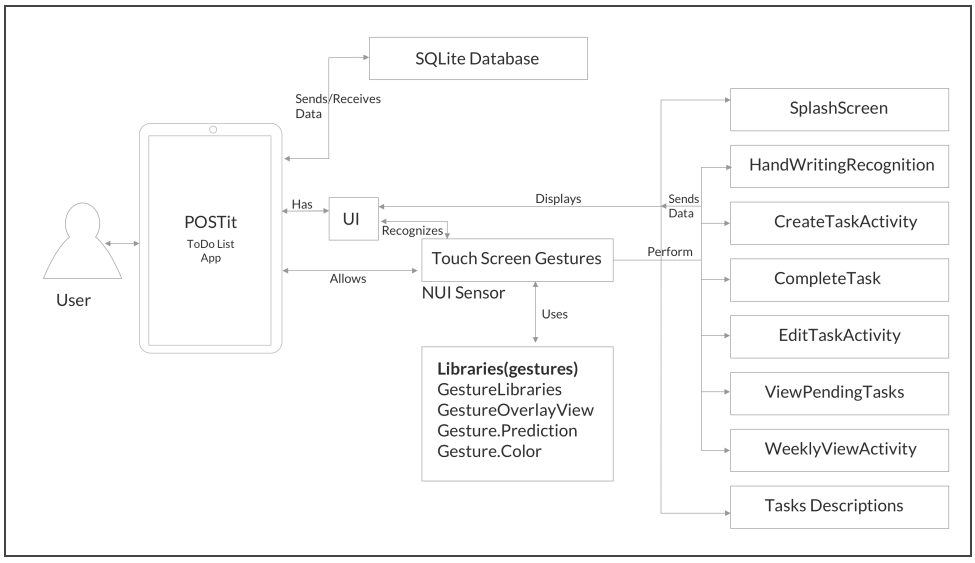
1.3.3 Change 3: Inclusion of handwriting recognition

- From our user study and research, we realized that some people prefer the traditional method of writing down the tasks over typing it out. Hence, we decided to add handwriting recognition to our to-do list app.
- Moreover, this platform is reflective of the human experience. Users that are not familiar with typing would prefer to write it down and save it.
- The add task page has a pen icon which redirects the user to another page. This page can be used as a scratchpad, where the user can write down the name of the task to be done using their finger.

- Since these are all smaller words and symbols we wouldn't have to worry about fatigue.

2 Final Architecture

2.1 System Architecture



2.2 Code Modules

The various classes and its methods are documented as follows:

1. **class createTaskActivity:** This class is used to create a new task. It has the following methods:
   - protected void onCreate(Bundle savedInstanceState): This method creates a new object for the gesture and compares the gesture drawn on the screen to the pre-defined gesture set. It also sets up the display and assigns default date and time.
   - public boolean onCreateOptionsMenu(Menu menu): This method inflates the screen into another screen with various menu bars regarding various details about the tasks.
   - private void getVars(): This is a getter method that is used to get the values of date, month, year and time for the task completion.
   - public void insertDb(): This method is used to insert various values of a task (i.e name, type, date, and location) to the database.
   - public void onClick(View v): This method is used to set values of the due date (date, month, and year) to values assigned by the user.

- public void onGesturePerformed function: This method identifies the gestures. It then confirms if the task name is assigned. If not, it gives out a message saying “Task name empty”, otherwise it inserts the task into the database.

2. **class HandWritingActivity:** This class mainly contains inbuilt functions that allow the user to write the task name using their finger which is then converted to text format.

   private class CandidateTag: This class succeeds in the handwritingActivity class and contains methods to add functionality to different buttons on the page providing handwriting recognition functionality.

- public void onDoneButtonClick(View v): This method is used to assign the text written by the user in the text bar as a task name.
- public void onClearButtonClick(View v): This method deletes whatever the user has written in the text bar.
- public void onDeleteButtonClick(View v): This method deletes that last (current) letter from what the user has written in the text bar. It allows the delete button to act as the backspace.
- public void onPencilButtonClick(View v): This method allows the user to choose the pen-type to write in the text bar. The user can choose among felt pen, fountain pen, calligraphic brush, calligraphic quill, and qalam.

3. **class MainActivity:** This class has methods that recognize the gesture drawn on the screen and makes method calls to methods in different classes for performing various functions on the tasks.

- private void updateUI(String selectQuery): This method takes the query as a parameter and updates the UI which displays the list of tasks.
- public void onGesturePerformed(): This method identifies the gesture and then filters the tasks accordingly i.e if the gesture is ‘p’ all the tasks of type ‘personal’ are pulled from the database if the gesture is ‘w’ all the tasks of type ‘work’ are pulled from the database.

5. **class ViewTaskActivity:** This method is used to display all the details of a task i.e its name, type, due date, and location.

- private void getTaskDetails(): This getter method is used to get various details of a task like its type(personal/work), due date, and time and location using string matching.
- public void onGesturePerformed(): This method identifies the gesture if the gesture is sigma, it provides the option to edit the already existing tasks, if the gesture is an S, it saves the gesture, and if the gesture is a left curved swipe then we go back to the previous page.

6. **class Item**: This is a getter class with only the getter methods. These methods are used in various classes for getting the values of the task title, task type, and task due date.

- public String getTitle(): Used to get the name of the tasks.
- public String getType(): Used to get the type of the tasks
- public String getDate(): Used to get the due date for the task.

7. **public class WeeklyViewActivity**: It is used to display the tasks to be performed per week.

- private void getviews(): Sets the layout for different days of the week.
- private void getVars(): This method retrieves the tasks from the database and creates an arrayList which is later used to list the tasks for a weekly basis.
- private void getTaskDetails(): This method gets the details of various tasks from the database and displays the values.

8. **public class EditTaskActivity:** This class is used to Edit the tasks already created.

- private void getTaskDetails(): This is a getter method that helps to retrieve all values for the original task.
- public void onClick(View v): This method allows us to set a different value to the due date, and time.
- public void updateDB(): This method updates the edited values in the database.

2.3 Third-Party Tools:

- **Sketch:** It is a design toolkit that was used to design the prototype of the app interface. Later these designs were imported to Android Studio for the front-end development. https://www.sketch.com/
- **Android Studio:** The main IDE(Integrated Development Environment) used to code, debug and run the app. Various frameworks and libraries could be run on it.

https://developer.android.com/studio

- **Android SDK:** The software development kit contains various tools like emulators and ADB bridge that were used to make the Android program execute with Java.
  https://developer.android.com/studio
- **Gradle:** It is a build-automation tool that was used to build our app and run it on the JVM in Android Studio.
  https://docs.gradle.org/current/userguide/userguide.html
- **SQLite:** The RDBMS (Relational Database Management System) was used to store and manage the data of the tasks from the users.
  https://www.sqlite.org/index.html

2.4 Source Code

- https://github.com/apoorva7597/To-do-List-app-with-Handwriting-Recognition

3 Future Work

We would like to make the app to be more user friendly by making the following changes:

- In the future, we would like to add the speech to text feature so that the user can verbally interact with the app as well. This could be developed using DialogFlow. This makes the application accessible for the visually impaired users as well.
- Additionally, the app could be scaled to a tablet platform for iOS and Android. Users generally do not use the virtual keyboard in the case of a tablet. Since tablets have bigger screen sizes and a stylus, they could be more comfortable for the users in the terms of writing out their tasks which will be then converted to typewritten text using our handwriting recognition feature.
- For now, we only have handwriting recognition for the name of the task. We would want to extend this feature for date, time and location as well.
- Furthermore, we could implement a daily task option and a reminder notification that the user receives for various tasks that need to be done.
- Before the prototype becomes a viable application we would need to evaluate it with more users and also ensure the accuracy is above a certain threshold for all interactions.