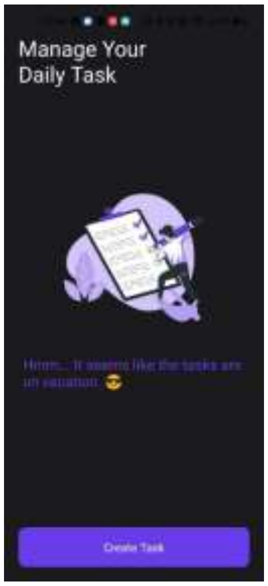
Todo-list Assignment

عبدالرحمن احمد ابراهيم

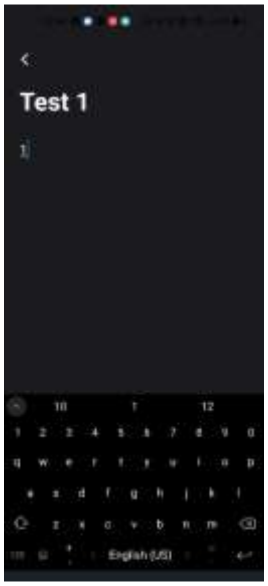
Sec 1

1 – Screens



- Homepage where at start of application there is no tasks so there is a text, button to add task and image

- after clicking on create task button it shows a hidden view where user can write name for task and description

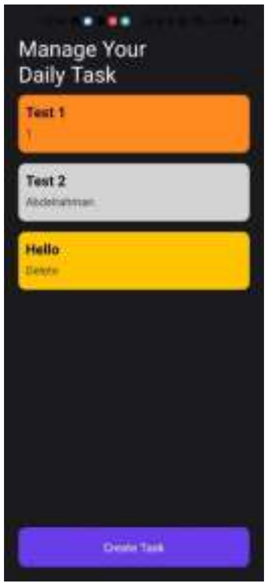


After that the user click go back button so that he can see all notes he added

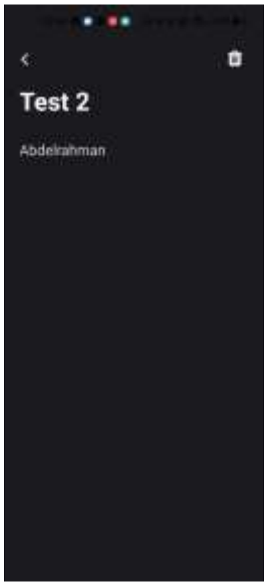
Main page where all the tasks that the user add exist

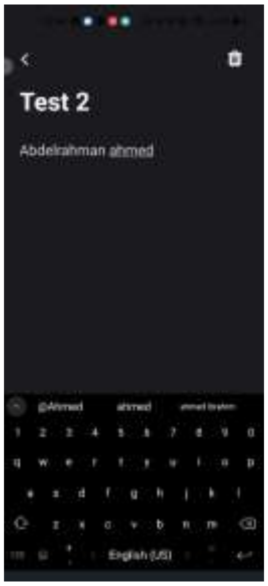
User can click on the note he wants to edit

Every note is given a random color from array that contains some predefined colors



After that view appears so the user can edit or delete the note





Colors and fonts

Note colors :

```
    '#4D90FE', // Soft Blue
    '#FFCC00', // Mellow Yellow
    '#FF6B6B', // Warm Coral
    '#A8E6CF', // Pastel Green
    '#D3D3D3', // Light Gray
    '#FF8225',
```

Background color : `#1A1A1D`

Button color : `#1A1A1D`

Font used : `Roboto`

Style

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#1A1A1D',
    justifyContent: 'flex-start',
    paddingHorizontal: 20,
  },
  MainText: {
    color: 'white',
    fontFamily: 'Roboto',
    fontSize: 30,
    fontWeight: 100,
    marginTop: 40,
    marginBottom: 10,
  },
  notesList: {
    width: '100%',
    height: '75%',
  },
  note: {
    backgroundColor: '#FF8225',
    borderRadius: 10,
    minHeight: 80,
    padding: 10,
    marginBottom: 15,
    width: '100%',
  },
  noteName: {
    color: 'black',
    fontFamily: 'Roboto',
    fontSize: 20,
    fontWeight: 'bold',
    marginBottom: 5,
  },
  noteDescription: {
    color: '#343131',
    fontFamily: 'Roboto',
    fontSize: 16,
    fontWeight: 100,
  },
  createTaskButton: {
    backgroundColor: '#672EE3',
    width: '100%',
    height: 55,
    borderRadius: 10,
    alignItems: 'center',
    justifyContent: 'center',
    position: 'absolute',
    bottom: 20,
    left: 20,
  },
  buttonText: {
    color: 'white',
    fontSize: 16,
  },

  addTaskWindow: {
    alignSelf: 'center',
    position: 'absolute',
    width: '110%',
    zIndex: 10,
    backgroundColor: '#1A1A1D',
    height: '100%',
    marginTop: 50,
  },
  addTaskWindow_buttons: {
    flexDirection: 'row',
    justifyContent: 'space-between',
    width: '100%',
    height: 50,
    alignItems: 'center',
  },
  addTaskWindow_createTaskBtn: {
    width: 50,
    height: 50,
    borderRadius: '50%',
    paddingLeft: 15,
    alignItems: 'flex-start',
    justifyContent: 'center',
  },
  addTaskWindow_deleteTaskBtn: {
    width: 50,
    height: 50,
    borderRadius: '50%',
    paddingTop: 0,
    alignItems: 'flex-start',
    justifyContent: 'center',
  },
  nameInput: {
    color: 'white',
    fontFamily: 'Roboto',
    fontSize: 35,
    fontWeight: 'bold',
    marginBottom: 5,
    outlineStyle: 'none',
    paddingHorizontal: '5%',
  },
  descriptionInput: {
    color: '#dedede',
    fontFamily: 'Roboto',
    fontWeight: 100,
    height: 550,
    marginTop: 5,
    outlineStyle: 'none',
    paddingHorizontal: '5%',
  },

  emptyContainer: {
    flex: 1,
    justifyContent: 'flex-start',
    alignItems: 'center',
    height: 600,
    width: '100%',
    marginTop: 100,
  },
  emptyImage: {
    width: 250,
    height: 250,
    marginBottom: 10,
    resizeMode: 'contain',
  },
  emptyText: {
    color: '#672EE3',
    fontFamily: 'arial',
    fontSize: 20,
    height: 50,
    fontWeight: 100,
  },
});
```

Logic and code for adding and deleting component

```
const handleSubmit = () => {
    if (name === '') {
      toggleVisibility();
      return;
    } else {
      const taskData = {
        id: taskId.toString(),
        name: name,
        description: description,
        color: getRandomColor(),
      };

      setNotes([...notes, taskData]);
      // alert(taskData.name);
      setTaskId(taskId + 1);

      setName('');
      setDescription('');
      toggleVisibility();
    }
  };

  const previewSelectedNote = (item) => {
    setName(item.name);
    setDescription(item.description);
    togglePreviewVisibility();
  };

  const handleModify = () => {
    if (
      selectedTask.name === name &&
      selectedTask.description === description
    ) {
      togglePreviewVisibility();
    } else {
      const updatedNotes = notes.map((item) =>
        item.id === selectedTask.id ? { ...item, name, description } : item
      );
      setNotes(updatedNotes); // Update the notes state with the modified note
      setName('');
      setDescription('');
      togglePreviewVisibility();
    }
  };

  const handleDelete = (id) => {
    const updatedNotes = notes.filter((note) => note.id !== id);
    setNotes(updatedNotes);
    setName('');
    setDescription('');
    togglePreviewVisibility();
  };


                 <FlatList
            style={styles.notesList}
            data={notes}
            keyExtractor={(item) => item.id}
            renderItem={({ item }) => (
              <TouchableOpacity
                onPress={() => {
                  previewSelectedNote(item);
                  setSelectedTask(item);
                }}
                style={[styles.note, { backgroundColor: item.color }]}>
                <Text style={styles.noteName}>{item.name}</Text>
                <Text style={styles.noteDescription}>{item.description}</Text>
              </TouchableOpacity>
            )}
          />
```