Lesson 4

Styling, Layout & Lists

Lesson Objectives

Understand styling in React Native

Use Flexbox for layout

Build scrollable layouts

Render lists using FlatList

Styling in React Native

How Styling Works

No CSS files

No class selectors

Styles are JavaScript objects

```
<Text style={{ color: 'blue', fontSize: 20 }}>
    Hello
</Text>
```

Using StyleSheet (Recommended)

```
import { StyleSheet } from 'react-native';

const styles = StyleSheet.create({
  title: {
    fontSize: 24,
    fontWeight: 'bold',
  },
});
```

<Text style={styles.title}>React Native</Text>

Common Style Properties

Color, fontSize, fontWeight, backgroundColor, padding, margin, borderRadius

Units in React Native

Numbers only

NO px, %, em

fontSize: 16

Flexbox Layout (Core Concept)

React Native uses Flexbox by default

```
<View style={{ flex: 1 }}>
```

Main Flexbox Properties:

- row | column
- justifyContent
- alignItems

Example:

Row

```
<View style={{ flexDirection: 'row' }}>
  <Text>Left</Text>
  <Text>Right</Text>
</View>
```

Centered Content

```
14
15  <View style={styles.container}>
16    <Text>Hello</Text>
17  </View>
18
19  container: {
20    flex: 1,
21    justifyContent: 'center',
22    alignItems: 'center',
23  }
```

What is ScrollView?

Scrollable container

Good for small content

Renders everything at once

```
import { ScrollView, Text } from 'react-native';

<ScrollView>
  <Text>Item 1</Text>
  <Text>Item 2</Text>
  <Text>Item 3</Text>
</ScrollView>
```

When NOT to Use ScrollView

Large Lists

- Hundreds of items
- When thats the case use FlatList(!important)

What is FlatList?

- Optimized list component
- Renders only visible items
- High performance

FlatList Basic Example

```
import { FlatList, Text } from 'react-native';

const data = [
  { id: '1', name: 'Student 1' },
  { id: '2', name: 'Student 2' },
];

<FlatList
  data={data}
  keyExtractor={(item) => item.id}
  renderItem={({ item }) => (
    <Text>{item.name}</Text>
  )}
/>
```

FlatList vs ScrollView

ScrollView	FlatList
Small content	Large lists
Renders all items	Renders only visible
Simple	Optimized

Styling FlatList Items

```
renderItem={({ item }) => (
  <View style={styles.card}>
    <Text>{item.name}</Text>
  </View>
)}
```

```
card: {
  padding: 15,
  margin: 10,
  backgroundColor: '#eee',
  borderRadius: 8,
}
```

Mini App

```
const students = [
  { id: '1', name: 'Aida' },
  { id: '2', name: 'Bek' },
  { id: '3', name: 'Nursultan' },
];

<FlatList
  data={students}
  renderItem={({ item }) => (
    <View style={styles.card}>
      <Text>{item.name}</Text>
    </View>
  )}
/>
```

Homework Assignment

- API CRUD Operations
- POSTMAN
- NodeJS + ExpressJS
- NeonTech

ToDO

Screens:

- Home
  - Add new (action)
  - Delete
  - Edit
  - Mark(completed/ not completed)
  - Search
- Profile:
  - ProfileCard
  - Log out
  - History ToDo

Backend:

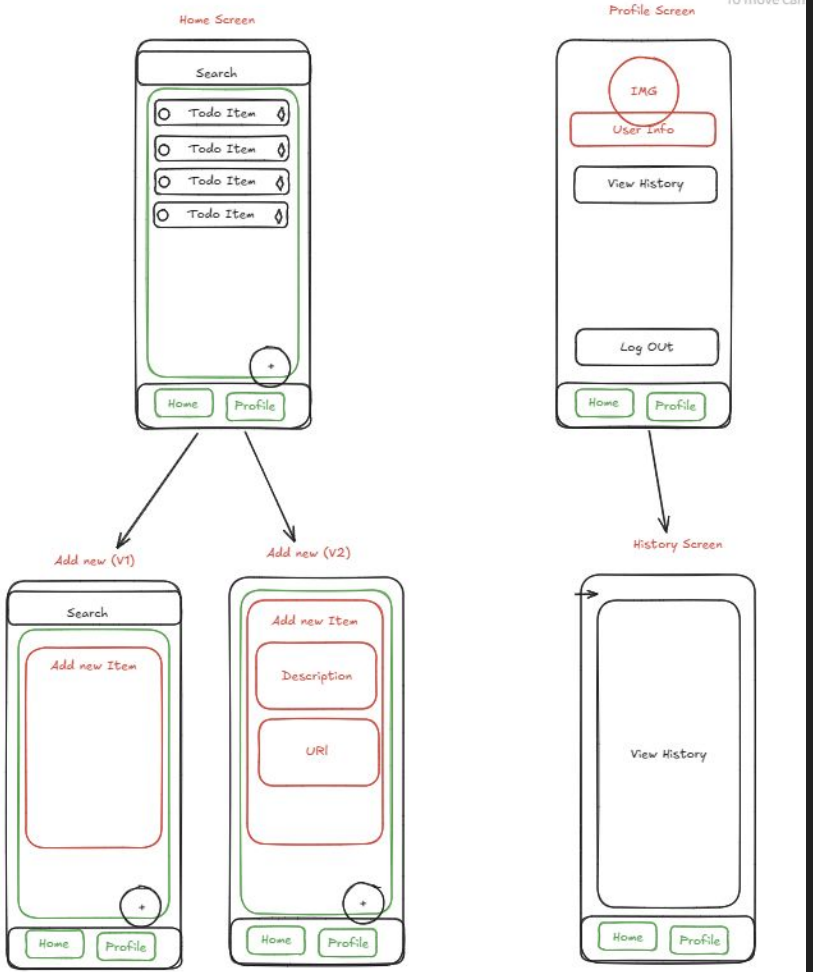
- API(Neon Tech)
- `NodeJS + ExpressJS
- Python
- Java
- c++

API:

Create DB

Api endpoint (CRUD(https:todo.app/api/create))

POSTMAN


Home Screen
Search
Todo Item
Todo Item
Todo Item
Todo Item
Home
Profile
Profile Screen
IMG
User Info
View History
Log OUt
Home
Profile
Add new (V1)
Search
Add new Item
Home
Profile
Add new (V2)
Add new Item
Description
URl
Home
Profile
History Screen
View History
Home
Profile