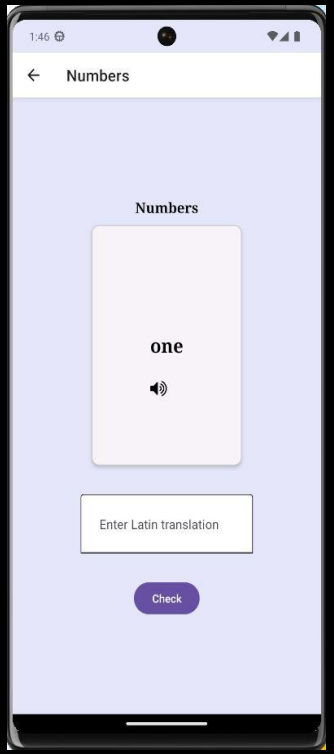
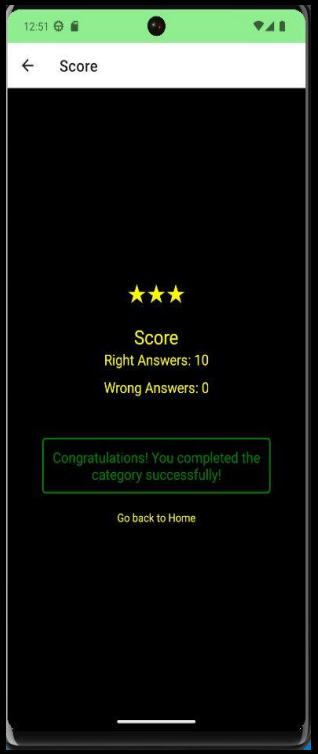
Latin-To-Go : DEMO



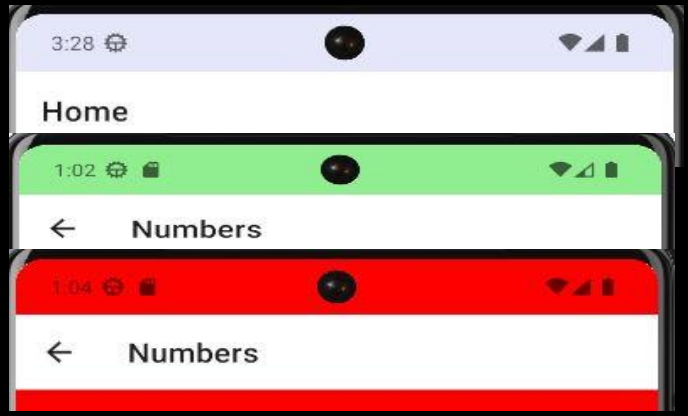


Expo-komponenter

StatusBar:

```
import { StatusBar } from "expo-status-bar";
```

```
useEffect(() => {
  StatusBar.setBackgroundColor(backgroundColor);

  return () => {
    StatusBar.setBackgroundColor("lavender");
  };
}, [backgroundColor]);
```



Haptics:

```
import * as Haptics from "expo-haptics";
```

```
const triggerWrongAnswerVibration = async () => {
  await Haptics.selectionAsync();
};

const checkAnswer = () => {
  setIsFlipped(true);
  const currentUserAnswer = userAnswer[currentCardIndex];

  if (
    currentUserAnswer.toLowerCase() ===
    cards[currentCardIndex].latin.toLowerCase()
  ) {
    setFeedback("Correct!");
  } else {
    setFeedback("Wrong!");
    triggerWrongAnswerVibration();
  }
```

Audio:

FlashcardScreenComponents.tsx

```tsx
import { Audio } from "expo-av";
```

```tsx
const playLatinWordAudio = async (latinWord: LatinWord) => {
  try {
    const audioPath = audioMapping[latinWord];

    if (audioPath) {
      const { sound } = await Audio.Sound.createAsync(audioPath);

      await sound.playAsync();
    } else {
      console.error(`Audio file not found for Latin word: ${latinWord}`);
    }
  } catch (error) {
    console.error("Error playing audio", error);
  }
};
```

```tsx
<View style={styles.cardTop}>
  <TouchableOpacity
    onPress={() =>
      playLatinWordAudio(cards[currentCardIndex].latin)
    }
    style={styles.soundIcon}
  >
    <FontAwesome name="volume-up" size={24} color="black" />
  </TouchableOpacity>
</View>
```

assets

AudioMapping.ts

```ts
const Decem = require("../assets/Decem.mp3");
const Duo = require("../assets/Duo.mp3");
const Novem = require("../assets/Novem.mp3");
const Octo = require("../assets/Octo.mp3");
const Quattuor = require("../assets/Quattuor.mp3");
const Quinque = require("../assets/Quinque.mp3");
const Septem = require("../assets/Septem.mp3");
```

```ts
const audioMapping = {
  unus: Unus,
  duo: Duo,
  tres: Tres,
  quattuor: Quattuor,
  quinque: Quinque,
```

```ts
export type LatinWord = keyof typeof audioMapping;
export default audioMapping;
```

Notifications:

```
import AsyncStorage from "@react-native-async-storage/async-storage";
import * as Notifications from "expo-notifications";

const generateDeviceId = async () => {
  let deviceId = await AsyncStorage.getItem("device_id");

  if (!deviceId) {
    deviceId = Math.random().toString(36).substring(7);
    await AsyncStorage.setItem("device_id", deviceId);
  }

  return deviceId;
};
```

```
const sendCongratulatoryNotification = async (incorrectCount: number) => {
  let title, body;

  if (incorrectCount === 0) {
    title = "Congratulations!";
    body = "You completed the category successfully!";
  } else if (incorrectCount <= 5) {
    title = "Good job...almost there!";
    body = "You got some answers wrong, but you're making progress!";
  } else {
    title = "You have to keep practicing!";
    body = "You got quite a few answers wrong, but don't give up!";
  }

  sendNotification(title, body);
};
```

```
export const getCongratulatoryMessage = (incorrectCount: number) => {
  let message = "";

  if (incorrectCount === 0) {
    message = "Congratulations! You completed the category successfully!";
  } else if (incorrectCount <= 5) {
    message = "You got some answers wrong, but you're making progress!";
  } else {
    message = "You got quite a few answers wrong, but don't give up!";
  }

  return message;
};

export default {
  sendCongratulatoryNotification,
  getCongratulatoryMessage,
};
```

React Native - komponenter

FlashcardScreenComponents.tsx

```tsx
import { StyleSheet, Text, TouchableOpacity, View } from "react-native";
import { Button, Card, TextInput } from "react-native-paper";
```

Card:

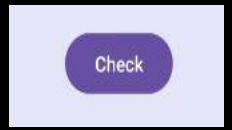
```tsx
<Card style={styles.card}>
  <TouchableOpacity onPress={() => setIsFlipped(!isFlipped)}>
    <Text style={styles.cardText}>
      {isFlipped
        ? cards[currentCardIndex].latin
        : cards[currentCardIndex].english}
    </Text>
  </TouchableOpacity>
  {!isFlipped && (
    <View style={styles.cardTop}>
      <TouchableOpacity
        onPress={() =>
          playLatinWordAudio(cards[currentCardIndex].latin)
        }
        style={styles.soundIcon}
      >
        <FontAwesome name="volume-up" size={24} color="black" />
      </TouchableOpacity>
    </View>
  )}
</Card>
```

```tsx
const goToNextCard = () => {
  if (currentCardIndex < cards.length - 1) {
    setCurrentCardIndex(currentCardIndex + 1);
    setIsFlipped(false);
    setFeedback(null);
  } else {
    let correctCount = 0;
    let incorrectCount = 0;

    for (let i = 0; i < cards.length; i++) {
      if (
        userAnswer[i] &&
        userAnswer[i].toLowerCase() === cards[i].latin.toLowerCase()
      ) {
        correctCount++;
      } else {
        incorrectCount++;
      }
    }
    if (correctCount === cards.length && incorrectCount === 0) {
      NotificationManager.sendCongratulatoryNotification();
      setAllAnswersCorrect(true);
    } else {
      setAllAnswersCorrect(false);
    }
    navigation.navigate("Score", {
      rightAnswers: correctCount,
      wrongAnswers: incorrectCount,
    });
  }
};
```

Button:

FlashcardScreenComponents.tsx

```
      {isFlipped && (
        <View style={styles.buttonContainer}>
          <Button mode="contained" onPress={goToNextCard}>
            Continue
          </Button>
        </View>
      )}

      {!isFlipped && (
        <View style={styles.buttonContainer}>
          <Button mode="contained" onPress={checkAnswer}>
            Check
          </Button>
        </View>
      )}
    </View>
  );
};
```



TouchableOpacity:

HomeScreen.tsx

```
<View style={styles.buttonContainer}>
  {circleButtons.map((button, index) => (
    <TouchableOpacity
      key={index}
      style={[styles.button, { backgroundColor: button.color }]}
      onPress={() => navigateToScreen(button.screen)}
    >
      <Text style={styles.buttonText}>{button.label}</Text>
    </TouchableOpacity>
  ))}
</View>
```

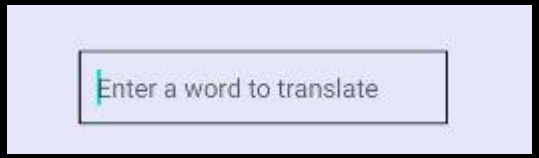
FlashcardScreenComponents.tsx

```
<TouchableOpacity onPress={() => setIsFlipped(!isFlipped)}>
  <Text style={styles.cardText}>
    {isFlipped
      ? cards[currentCardIndex].latin
      : cards[currentCardIndex].english}
  </Text>
</TouchableOpacity>
{!isFlipped && (
  <View style={styles.cardTop}>
    <TouchableOpacity
      onPress={() =>
        playLatinWordAudio(cards[currentCardIndex].latin)
      }
      style={styles.soundIcon}
    >
      <FontAwesome name="volume-up" size={24} color="black" />
    </TouchableOpacity>
```

TextInput:

HomeScreen.tsx

```tsx
<View style={styles.inputContainer}>
  <TextInput
    style={styles.input}
    placeholder="Enter a word to translate"
    onChangeText={(text) => setInputWord(text)}
    value={inputWord}
  />
</View>
```

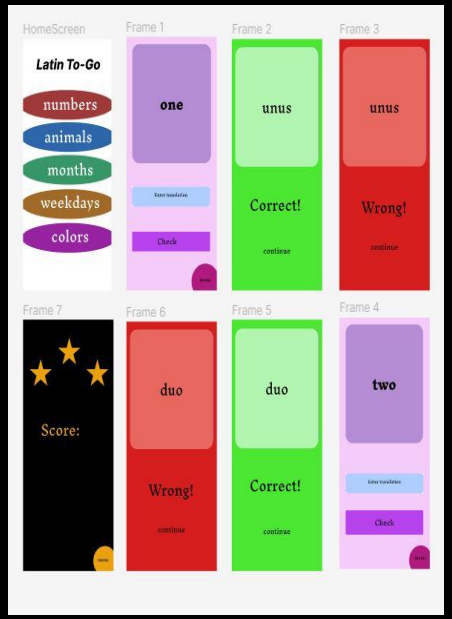


FlashcardScreenComponents.tsx

```tsx
<TextInput
  style={[styles.input, isFlipped && styles.disabledInput]}
  placeholder="Enter Latin translation"
  autoFocus={inputAutoFocus}
  onChangeText={(text) => {
    if (!isFlipped) {
      const updatedAnswers = [...userAnswer];
      updatedAnswers[currentCardIndex] = text;
      setUserAnswer(updatedAnswers);
    }
  }}
  value={userAnswer[currentCardIndex]}
  editable={!isFlipped}
/>
```

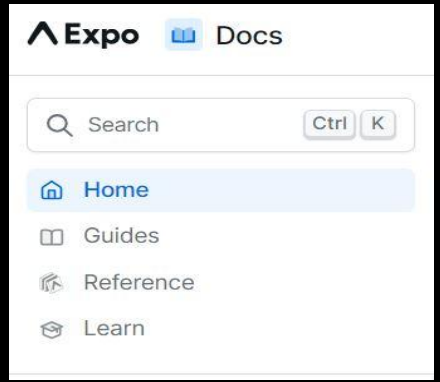
Planering:

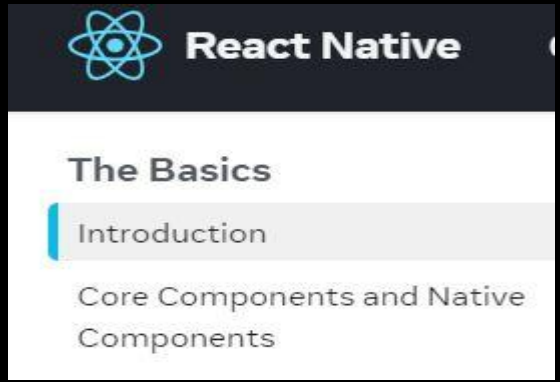
FIGMA



Genomförande:

Davids tutorial





Reflektioner:

- React-Native
  - Plattformsoberoende
  - Användarupplevelse
  - Document/Community
- React-Navigation
  - flexibel/anpassningsbar
- Expo
  - enklare/snabbare app-utveckling

Struktur:

```
Latin-To-Go/
  ├── assets/
  │   ├── audio1.mp3
  │   ├── audio2.mp3
  │   └── ...
  ├── components/
  │   └── FlashcardScreenComponents.tsx
  ├── config/
  │   └── AudioMapping.ts
  ├── screens/
  │   ├── FlashcardCategoryScreen.tsx
  │   ├── HomeScreen.tsx
  │   └── ScoreScreen.tsx
  ├── utils/
  │   └── NotificationManager.ts
  ├── cards/
  │   └── ConstantsData.ts
  ├── App.tsx
  ├── node_modules/
  ├── package.json
  ├── package-lock.json
  └── README.md
```

Tack för eran tid…