```
import { faCalendarAlt } from '@fortawesome/free-regular-svg-icons'
import { faDollarSign, faSmile, IconDefinition } from '@fortawesome/
free-solid-svg-icons'
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome'
import { MaterialUiPickersDate } from '@material-ui/pickers/typings/
date'
import 'date-fns'
import React, { useEffect, useState } from 'react'
import styled from 'styled-components'
import { updateGoal as updateGoalApi } from '../../../api/lib'
import { Goal } from '../../../api/types'
import { selectGoalsMap, updateGoal as updateGoalRedux } from
'../../../store/goalsSlice'
import { useAppDispatch, useAppSelector } from '../../../store/
hooks'
import DatePicker from '../../components/DatePicker'
import { Theme } from '../../components/Theme'
import { BaseEmoji } from 'emoji-mart'
import EmojiPicker from '../../components/EmojiPicker'
import GoalIcon from './GoalIcon'
import { TransparentButton } from '../../components/
TransparentButton'
import AddIconButton from './AddIconButton'

type Props = { goal: Goal };

const EmojiPickerContainer = styled.div<EmojiPickerContainerProps>`
  display: ${(props) => (props.isOpen ? 'flex' : 'none')};
  position: absolute;
  top: ${(props) => (props.hasIcon ? '10rem' : '2rem')};
  left: 0;
`;

// const AddIconButtonContainer =
styled.div<AddIconButtonContainerProps>`
// display: ${(props) => (props.shouldShow ? 'flex' : 'none')};

// `

// const GoalIconContainer = styled.div<GoalIconContainerProps>`
//   display: ${(props) => (props.shouldShow ? 'flex' : 'none')};
// `

export function GoalManager(props: Props) {
  const dispatch = useAppDispatch();

  const [emojiPickerIsOpen, setEmojiPickerIsOpen] = useState(false);


  const hasIcon = () => icon != null;

  const pickEmojiOnClick = (emoji: BaseEmoji, event:
React.MouseEvent) => {
    event.stopPropagation()
```

```
    setIcon(emoji.native)
    setEmojiPickerIsOpen(false)

    const updatedGoal: Goal = {
      ...props.goal,
      icon: emoji.native ?? props.goal.icon,
      name: name ?? props.goal.name,
      targetDate: targetDate ?? props.goal.targetDate,
      targetAmount: targetAmount ?? props.goal.targetAmount,
    }

    dispatch(updateGoalRedux(updatedGoal))

    // TODO(TASK-3) Update database
    updateGoalApi(props.goal.id, updatedGoal)
  }

  const [icon, setIcon] = useState<string | null>(null)

  useEffect(() => {
    setIcon(props.goal.icon || null);

  }, [props.goal.id, props.goal.icon])

  const addIconOnClick = (event: React.MouseEvent) => {
    event.stopPropagation()
    setEmojiPickerIsOpen(true)
  }

  const goal = useAppSelector(selectGoalsMap)[props.goal.id]

  const [name, setName] = useState<string | null>(null)
  const [targetDate, setTargetDate] = useState<Date | null>(null)
  const [targetAmount, setTargetAmount] = useState<number |
null>(null)



  useEffect(() => {
    setName(props.goal.name)
    setTargetDate(props.goal.targetDate)
    setTargetAmount(props.goal.targetAmount)
  }, [
    props.goal.id,
    props.goal.name,
    props.goal.targetDate,
    props.goal.targetAmount,
  ])

  useEffect(() => {
    setName(goal.name)
  }, [goal.name])
```

```
  const updateNameOnChange = (event:
React.ChangeEvent<HTMLInputElement>) => {
    const nextName = event.target.value
    setName(nextName)
    const updatedGoal: Goal = {
      ...props.goal,
      name: nextName,
    }
    dispatch(updateGoalRedux(updatedGoal))
    updateGoalApi(props.goal.id, updatedGoal)
  }

  const updateTargetAmountOnChange = (event:
React.ChangeEvent<HTMLInputElement>) => {
    const nextTargetAmount = parseFloat(event.target.value)
    setTargetAmount(nextTargetAmount)
    const updatedGoal: Goal = {
      ...props.goal,
      name: name ?? props.goal.name,
      targetDate: targetDate ?? props.goal.targetDate,
      targetAmount: nextTargetAmount,
    }
    dispatch(updateGoalRedux(updatedGoal))
    updateGoalApi(props.goal.id, updatedGoal)
  }

  const pickDateOnChange = (date: MaterialUiPickersDate) => {
    if (date != null) {
      setTargetDate(date)
      const updatedGoal: Goal = {
        ...props.goal,
        name: name ?? props.goal.name,
        targetDate: date ?? props.goal.targetDate,
        targetAmount: targetAmount ?? props.goal.targetAmount,
      }
      dispatch(updateGoalRedux(updatedGoal))
      updateGoalApi(props.goal.id, updatedGoal)
    }
  }

  return (
    <>


      <GoalManagerContainer>
        <NameInput value={name ?? ''}
onChange={updateNameOnChange} />

        <Group>
          <Field name="Target Date" icon={faCalendarAlt} />
          <Value>
            <DatePicker value={targetDate}
onChange={pickDateOnChange} />
          </Value>
```

```
        </Group>

        <Group>
          <Field name="Target Amount" icon={faDollarSign} />
          <Value>
            <StringInput value={targetAmount ?? ''}
onChange={updateTargetAmountOnChange} />
          </Value>
        </Group>

        <Group>
          <Field name="Balance" icon={faDollarSign} />
          <Value>
            <StringValue>{props.goal.balance}</StringValue>
          </Value>
        </Group>

        <Group>
          <Field name="Date Created" icon={faCalendarAlt} />
          <Value>
            <StringValue>{new
Date(props.goal.created).toLocaleDateString()}</StringValue>
          </Value>
        </Group>
      </GoalManagerContainer>

      <EmojiPickerContainer
        isOpen={emojiPickerIsOpen}
        hasIcon={hasIcon()}
        onClick={(event) => event.stopPropagation()}
      >
        <EmojiPicker onClick={pickEmojiOnClick} />
      </EmojiPickerContainer>

      <AddIconButtonContainer shouldShow={!hasIcon()} hasIcon={!
hasIcon()}>
        <TransparentButton onClick={addIconOnClick}>
          <FontAwesomeIcon icon={faSmile} size="2x" />
          <AddIconButtonText>Add icon</AddIconButtonText>
        </TransparentButton>
      </AddIconButtonContainer>

      <AddIconButton hasIcon={hasIcon()} onClick={addIconOnClick} />


      <GoalIconContainer shouldShow={hasIcon()}>
        <GoalIcon icon={goal.icon || null}
onClick={addIconOnClick} />
      </GoalIconContainer>



    </>
  )
```

```
}

type FieldProps = { name: string; icon: IconDefinition }
type AddIconButtonContainerProps = { shouldShow: boolean; hasIcon:
boolean }
type GoalIconContainerProps = { shouldShow: boolean }
type EmojiPickerContainerProps = { isOpen: boolean; hasIcon:
boolean }

const Field = (props: FieldProps) => (
  <FieldContainer>
    <FontAwesomeIcon icon={props.icon} size="2x" />
    <FieldName>{props.name}</FieldName>
  </FieldContainer>
)

const GoalManagerContainer = styled.div`
  display: flex;
  flex-direction: column;
  justify-content: flex-start;
  align-items: flex-start;
  height: 100%;
  width: 100%;
  position: relative;
`


const Group = styled.div`
  display: flex;
  flex-direction: row;
  width: 100%;
  margin-top: 1.25rem;
  margin-bottom: 1.25rem;
`
const NameInput = styled.input`
  display: flex;
  background-color: transparent;
  outline: none;
  border: none;
  font-size: 4rem;
  font-weight: bold;
  color: ${({ theme }: { theme: Theme }) => theme.text};
`


const FieldName = styled.h1`
  font-size: 1.8rem;
  margin-left: 1rem;
  color: rgba(174, 174, 174, 1);
  font-weight: normal;
`
const FieldContainer = styled.div`
  display: flex;
  flex-direction: row;
  align-items: center;
  width: 20rem;
```

```
  svg {
    color: rgba(174, 174, 174, 1);
  }
`
const StringValue = styled.h1`
  font-size: 1.8rem;
  font-weight: bold;
`
const StringInput = styled.input`
  display: flex;
  background-color: transparent;
  outline: none;
  border: none;
  font-size: 1.8rem;
  font-weight: bold;
  color: ${({ theme }: { theme: Theme }) => theme.text};
`


const Value = styled.div`
  margin-left: 2rem;
`



const GoalIconContainer = styled.div<GoalIconContainerProps>`
  display: ${(props) => (props.shouldShow ? 'flex' : 'none')};
`;


const AddIconButtonContainer =
styled.div<AddIconButtonContainerProps>`
  display: ${(props) => (props.shouldShow ? 'flex' : 'none')};
`;

const AddIconButtonText = styled.span`
  margin-left: 0.6rem;
  font-size: 1.5rem;
  color: rgba(174, 174, 174, 1);
`;
```