import React, { useEffect, useState } from 'react'

import { useSelector } from 'react-redux'

import { useHistory, useParams } from 'react-router'

import { QuestionInfo } from '../../components/questions/common'

import {

    SquareAnswers,

    CircleAnswers,

    NotificationAnswers,

    KeyboardAnswers,

    SquareThemes,

    RectangleTheme,

} from '../../components/questions'

import { types } from '../../redux/initialStates/questions/questionTypes'

import { getQuestions } from '../../redux/slices/questionsSlice'

import {

    ampEvent,

    ampEventWithEventProperty,

    ampSetUserProperty,

} from '../../utils/amplitude'

import './styles.scss'

import { paramsToQuery } from '../../utils/paramsToQuery'

import { getFlow, getUserId } from '../../redux/slices/userSlice'

import { flowTypes } from '../../utils/flowTypes'

const VIEVPORT\_RESIZE\_TO\_PX = 0.01

const QuestionsScreen = () => {

    const { id } = useParams()

    const history = useHistory()

    const questions = useSelector(getQuestions)

    const user\_id = useSelector(getUserId)

    const flow = useSelector(getFlow)

    const activeQuestion = questions[id - 1]

    const [out, setOut] = useState(false)

    let vh

    const onWindowResize = () => {

        vh = window.innerHeight \* VIEVPORT\_RESIZE\_TO\_PX

        document.documentElement.style.setProperty('--vh', `${vh}px`)

    }

    useEffect(() => {

        if (id === '1') {

            setTimeout(() => {

                ampSetUserProperty('origin', window.location.origin)

                ampEvent('Onboarding\_shown')

            }, 0)

        }

        onWindowResize()

        window.addEventListener('resize', onWindowResize)

    }, [])

    const nextQuestion = () => {

        ampEventWithEventProperty('Question\_answered', { question: +id })

        const isLastQuestion = +id === questions.length

        if (!isLastQuestion) {

            if (flow === flowTypes.THEMES) {

                setOut(true)

            }

            setTimeout(() => {

                history.push({

                    pathname: `/questions/${+id + 1}`,

                    search: paramsToQuery({ user\_id }),

                })

                if (flow === flowTypes.THEMES) {

                    setOut(false)

                }

            }, 400)

            return

        }

        history.push({

            pathname: `/processing`,

            search: paramsToQuery({ user\_id }),

        })

    }

    const questionsController = () => {

        switch (activeQuestion.componentType) {

            case types.SQUARE\_ANSWERS: {

                return (

                    <SquareAnswers

                        question={activeQuestion}

                        nextQuestion={nextQuestion}

                    />

                )

            }

            case types.WIDGET\_CIRCLES\_ANSWERS: {

                return (

                    <CircleAnswers

                        question={activeQuestion}

                        nextQuestion={nextQuestion}

                    />

                )

            }

            case types.TEXT\_CIRCLES\_ANSWERS: {

                return (

                    <CircleAnswers

                        question={activeQuestion}

                        nextQuestion={nextQuestion}

                    />

                )

            }

            case types.NOTIFICATION\_ANSWERS: {

                return (

                    <NotificationAnswers

                        question={activeQuestion}

                        nextQuestion={nextQuestion}

                    />

                )

            }

            case types.WIDGET\_SQUARE\_ANSWERS: {

                return (

                    <SquareAnswers

                        question={activeQuestion}

                        nextQuestion={nextQuestion}

                    />

                )

            }

            case types.KEYBOARD\_ANSWERS: {

                return (

                    <KeyboardAnswers

                        question={activeQuestion}

                        nextQuestion={nextQuestion}

                    />

                )

            }

            case types.SQUARE\_THEMES: {

                return (

                    <SquareThemes question={activeQuestion} nextQuestion={nextQuestion} />

                )

            }

            case types.RECTANGLE\_THEMES: {

                return (

                    <RectangleTheme

                        question={activeQuestion}

                        nextQuestion={nextQuestion}

                    />

                )

            }

            default: {

                return <p>Component now found</p>

            }

        }

    }

    return (

        <div

            className={`questions ${

                flow === flowTypes.THEMES ? (out ? 'out' : 'in') : ''

            }`}

        >

            <QuestionInfo

                id={id}

                question={activeQuestion.question}

                width={activeQuestion.questionWidth}

            />

            <div className="questions-answers">{questionsController()}</div>

        </div>

    )

}

export { QuestionsScreen }