Ministerul Educaţiei, Tineretului şi Sportului al Republicii Moldova

Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică şi Microelectronică

Departamentul Informatica si Ingineria sistemelor

RAPORT

Lucrare de laborator nr.2

la Programarea aplicațiilor mobile

Tema: Organiser Mobile Application (DAILY PLANNER)

A efectuat: st. gr. FI-181

Damean Alxandra

A verificat: prof.

Rusu Cristian

Chişinău 2020

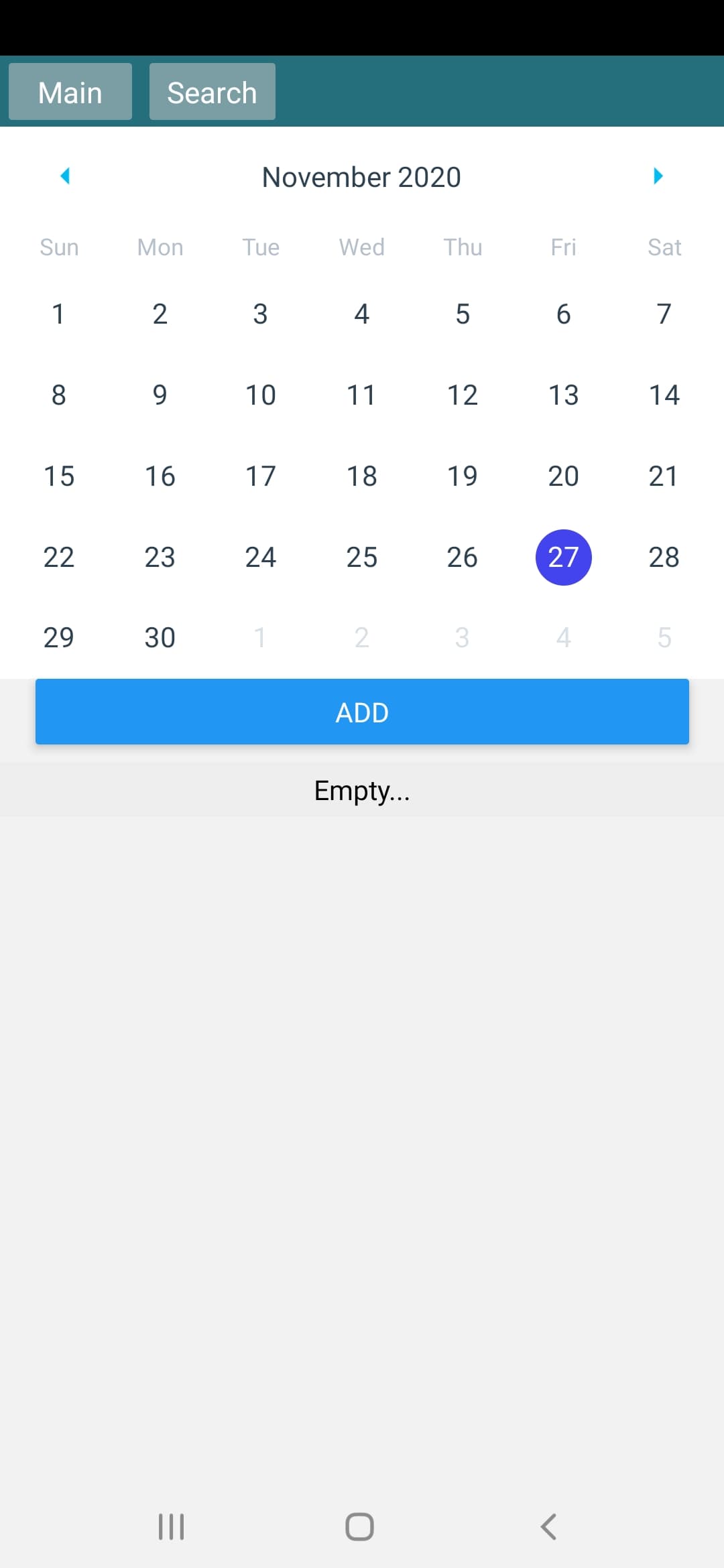
**Scopul lucrării de laborator:**

De realizat aplicație tip Organiser pe platforma aleasă.

La formarea și proiectarea aplicațiilor sunt introduse puncte stricte,

ce sunt obligatorii de respectat.

1. **MainActivity (structura/componente):**



const log = logger.createLogger()

log.setSeverity('debug')

function mapDispatchToProps(dispatch: Dispatch) {

  const actionCreators = { setSelectedDate, removeDateDescription, fetchDataFromXmlFileStorage }

  return bindActionCreators(actionCreators, dispatch)

}

function mapStateToProps(state: GlobalStateType) {

*// log['debug']('state: ', state)*

  return {

    selectedDateStr: state.calendarReducer.selectedDateStr,

    dates: state.calendarReducer.dates,

    isLoadedFirstTime: state.calendarReducer.isLoadedFirstTime,

  }

}

type MainScreenComponentPropType = ReturnType<typeof mapStateToProps> &

  ReturnType<typeof mapDispatchToProps> & {

    navigation: NavigationProp<ParamListBase>

  }

class MainScreenComponent *extends* Component<MainScreenComponentPropType> {

  constructor(props: MainScreenComponentPropType) {

    super(props)

    const initialDateStr: string = new Date().toISOString().slice(0, 10)

    this.props.setSelectedDate(initialDateStr)

  }

  componentDidMount(): void {

    if (!this.props.isLoadedFirstTime) {

      this.props.fetchDataFromXmlFileStorage()

    }

  }

  render(): ReactElement {

    const datesForCalendar = this.props.dates.reduce((result: any, element: any) => {

      result[element.dateStr] = { selected: true, selectedColor: '#9ee' }

      return result

*// eslint-disable-next-line @typescript-eslint/no-explicit-any*

    }, {} as any)

    const manualSelectedDate = this.props.selectedDateStr

      ? {

          [this.props.selectedDateStr]: { selected: true, selectedColor: '#44e' },

        }

      : {}

    const showedDates = this.props.dates.filter(

      (a: DateTimeType) => a.dateStr === this.props.selectedDateStr,

    )

    return (

      <ScrollView>

        <Calendar

*style*={styles.calendarStyle}

*onDayPress*={(day) => {

            log['debug'](`MainScreenComponent: Selected day: ${day.dateString}`)

            this.props.setSelectedDate(day.dateString)

          }}

*markedDates*={{

            ...datesForCalendar,

            ...manualSelectedDate,

          }}

        />

        <View *style*={styles.addButton}>

          <Button

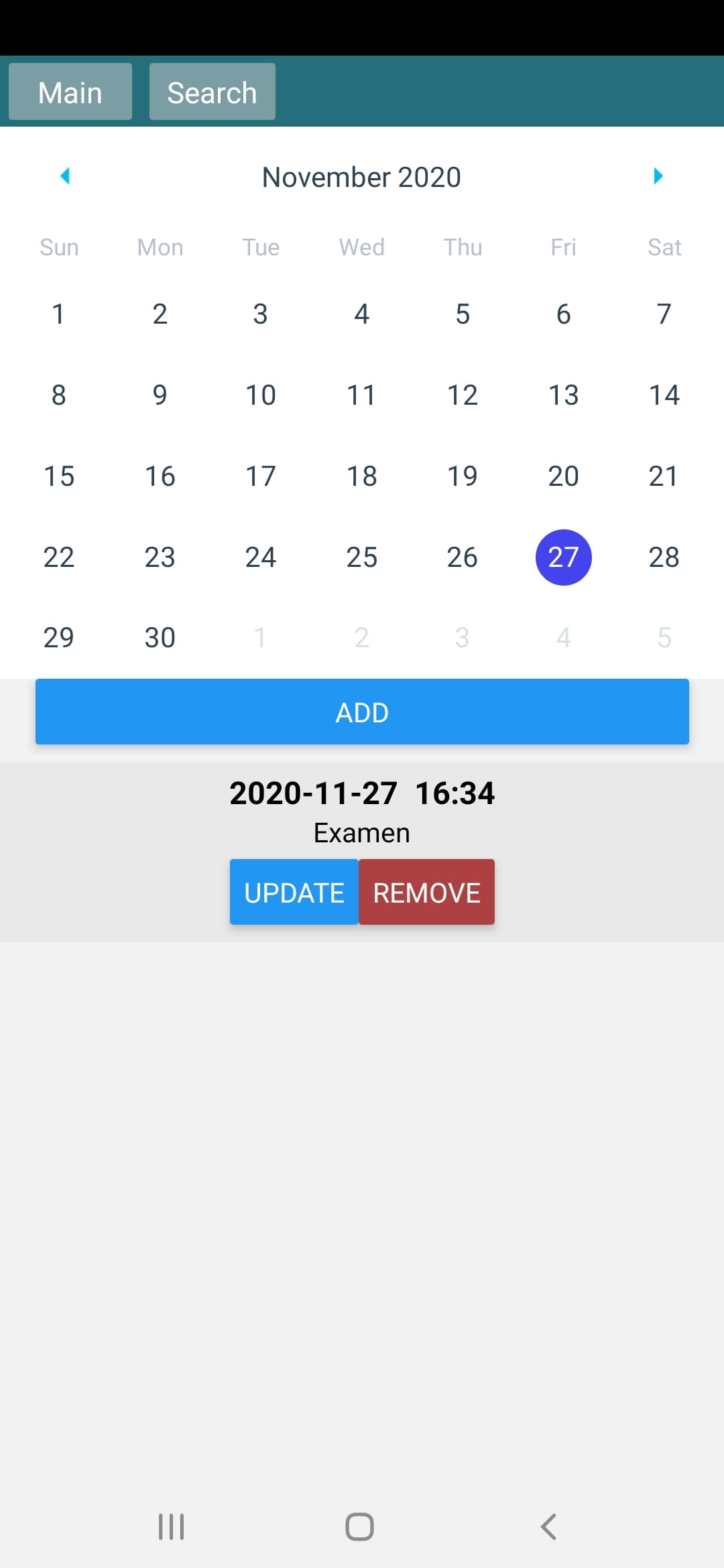
*onPress*={() => this.props.navigation.navigate(routeNames.AddScreen)}

*title*="Add"

          />

        </View>

* Buttons (Add/Remove/Update)



 <TextInput

*defaultValue*={this.state.dateTime.description}

*onChangeText*={(text) =>

              this.setState({ dateTime: { ...this.state.dateTime, description: text } })

            }

*multiline*={true}

*style*={styles.inputStyle}

          />

          <TimePicker

*hours*={this.state.dateTime.hours}

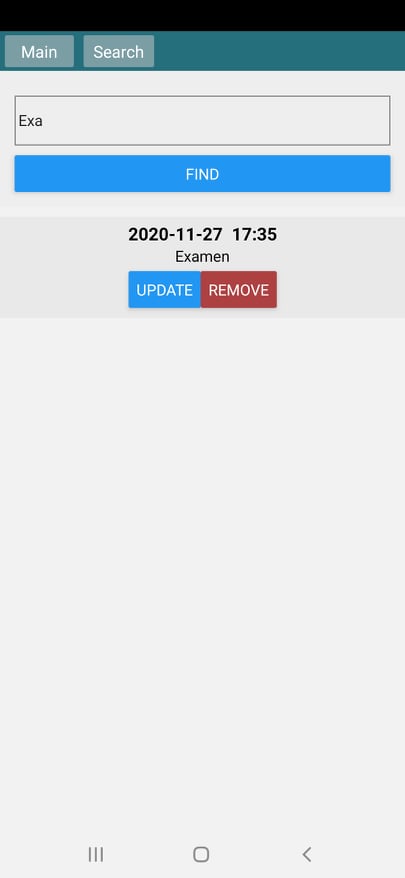
*minutes*={this.state.dateTime.minutes}

*onTimeChange*={this.onTimeChange}

          />

          <Button *onPress*={this.onUpdateDescriptionButtonClick} *title*={`Update description`} />

* Căutare (caută conform cuvintelor cheie)



  return (

    <ScrollView>

      <View *style*={styles.searchContainer}>

        <TextInput

*style*={styles.textInputStyle}

*defaultValue*={searchValue}

*onChangeText*={(text) => setSearchValue(text)}

        />

        <Button *onPress*={onFindClick} *title*="Find" />

      </View>

      {elements}

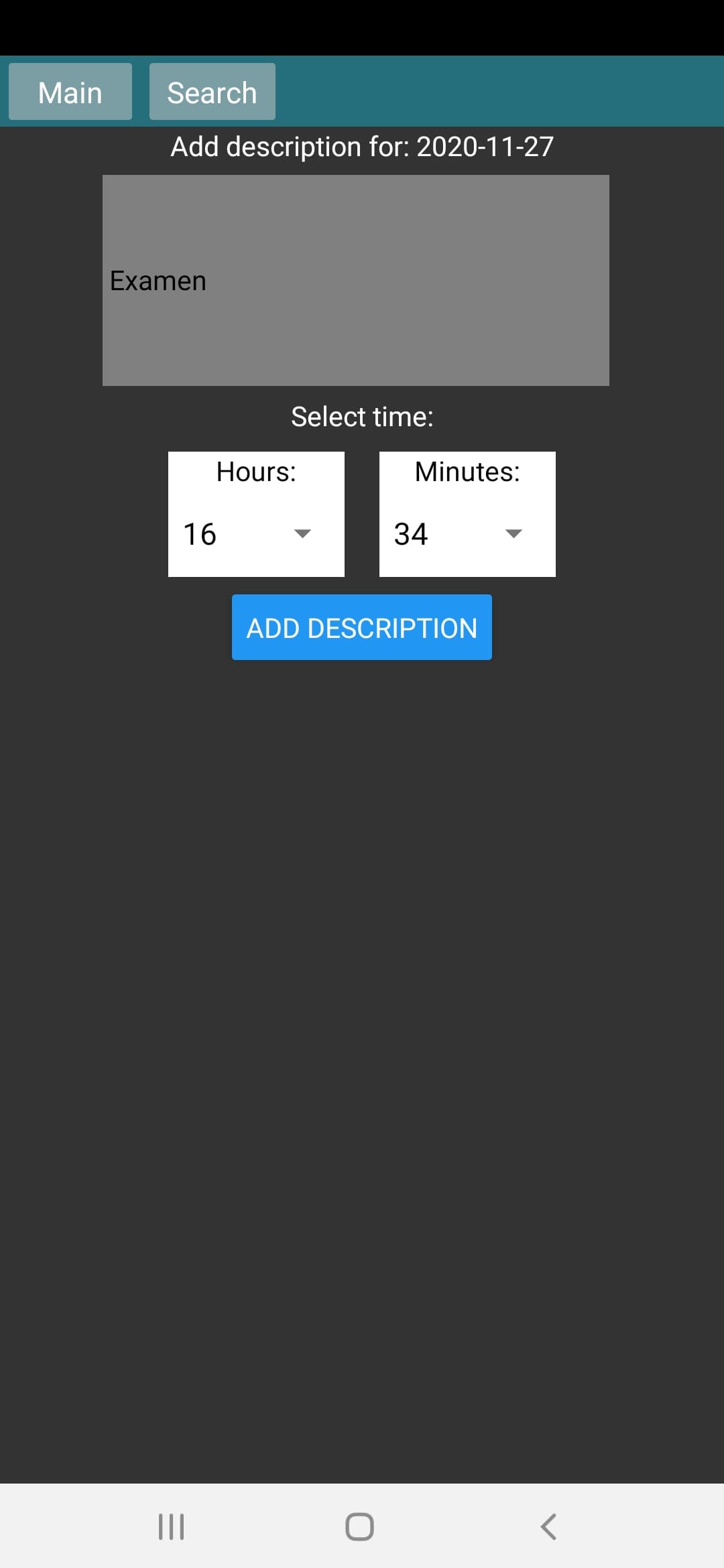
    </ScrollView>

  )

}

2.AddtActivity

* Data/Time controller
* Info TextBox
* Buttons și altele (la discreție conform specificului aplicației)



return (

    <Fragment>

      <Text *style*={{ color: 'white' }}>Select time:</Text>

      <View *style*={styles.container}>

        <View *style*={styles.timePicker}>

          <Text *style*={{ color: 'black', textAlign: 'center' }}>Hours:</Text>

          <Picker

*selectedValue*={props.hours}

*style*={{ height: 50, width: 100 }}

*onValueChange*={(itemValue) => setHours(Number(itemValue))}

          >

            {hours.map((item, index) => (

              <Picker.Item

*key*={index}

*label*={item < 10 ? '0' + String(item) : String(item)}

*value*={item}

              />

            ))}

          </Picker>

        </View>

        <View *style*={styles.timePicker}>

          <Text *style*={{ color: 'black', textAlign: 'center' }}>Minutes:</Text>

          <Picker

*selectedValue*={props.minutes}

*style*={{ height: 50, width: 100 }}

*onValueChange*={(itemValue) => setMinutes(Number(itemValue))}

          >

            {minutes.map((item, index) => (

              <Picker.Item

*key*={index}

*label*={item < 10 ? '0' + String(item) : String(item)}

*value*={item}

              />

            ))}

          </Picker>

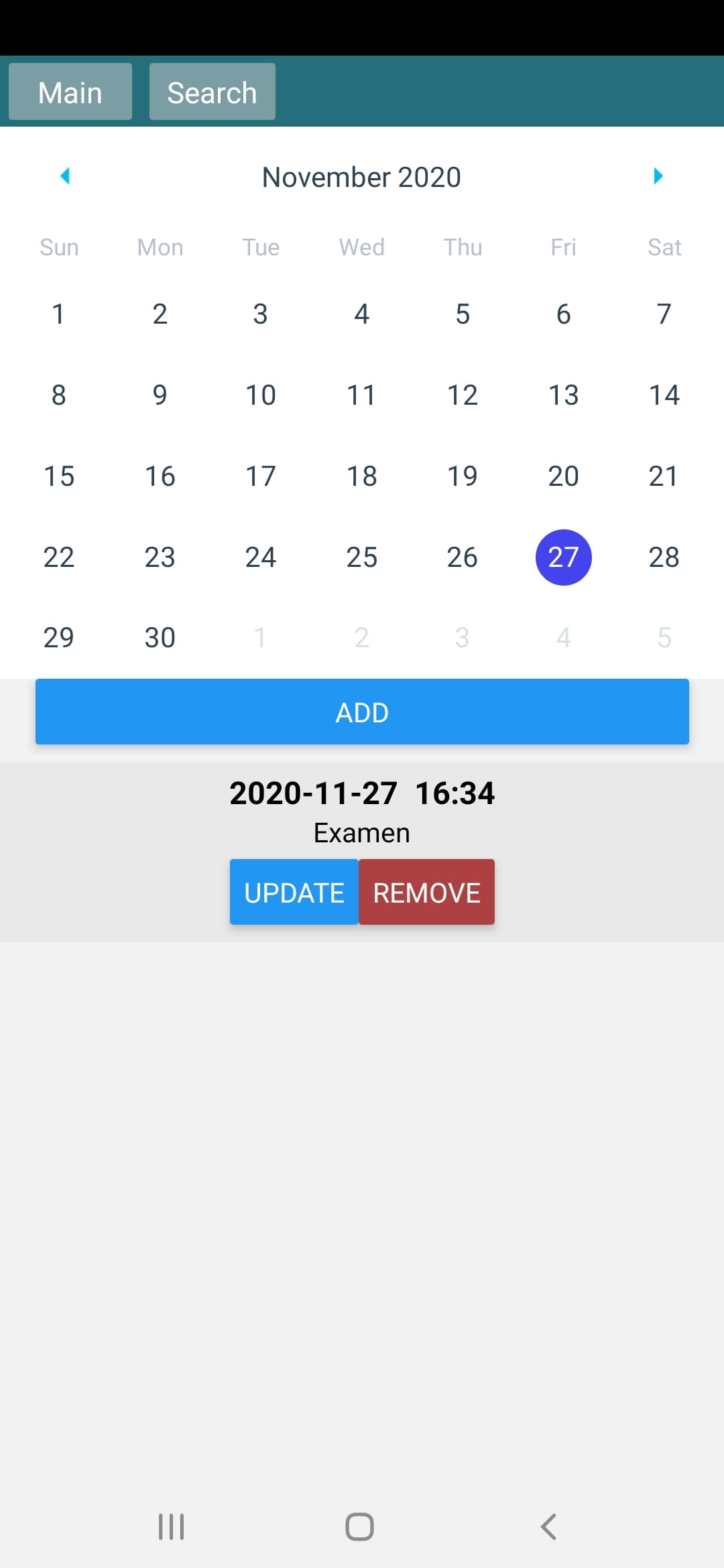
        </View>

      </View>

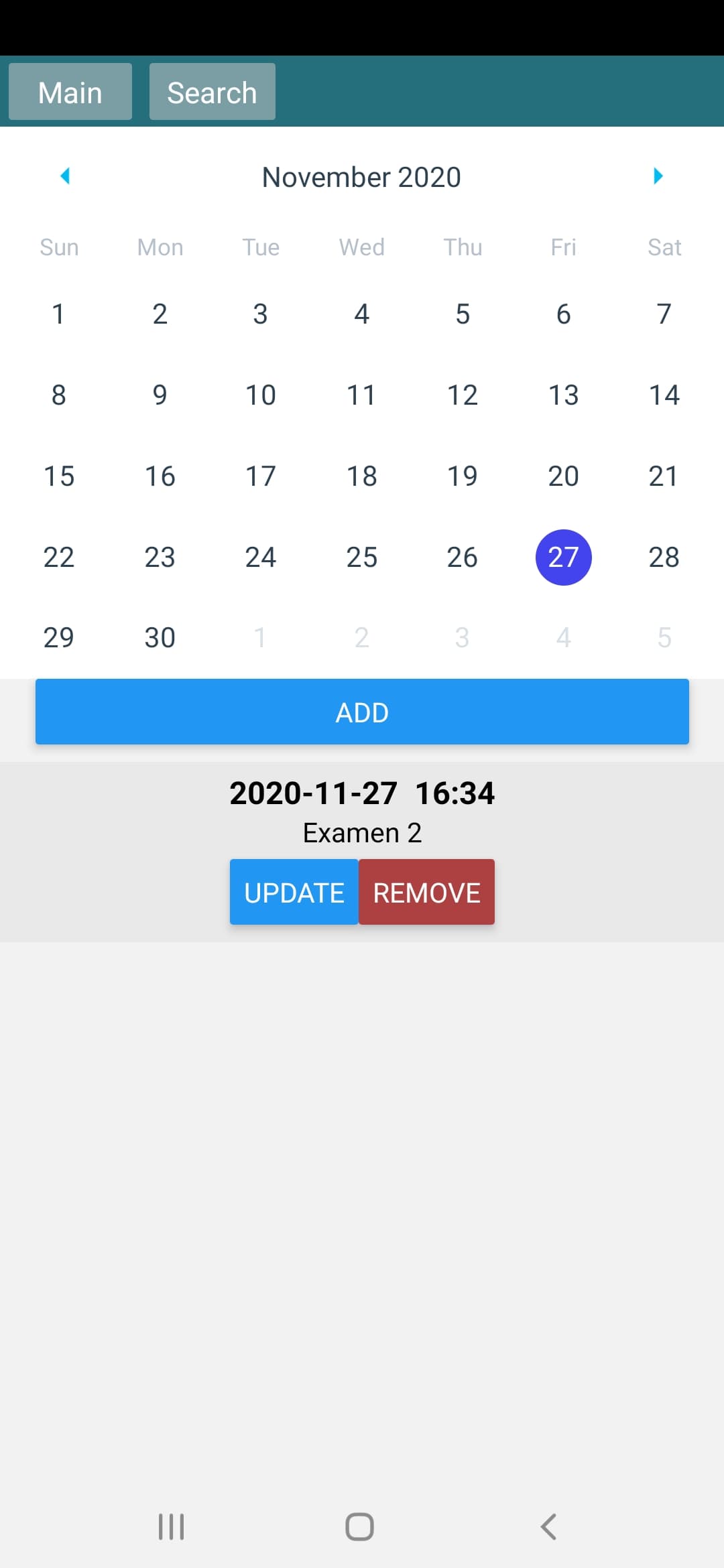
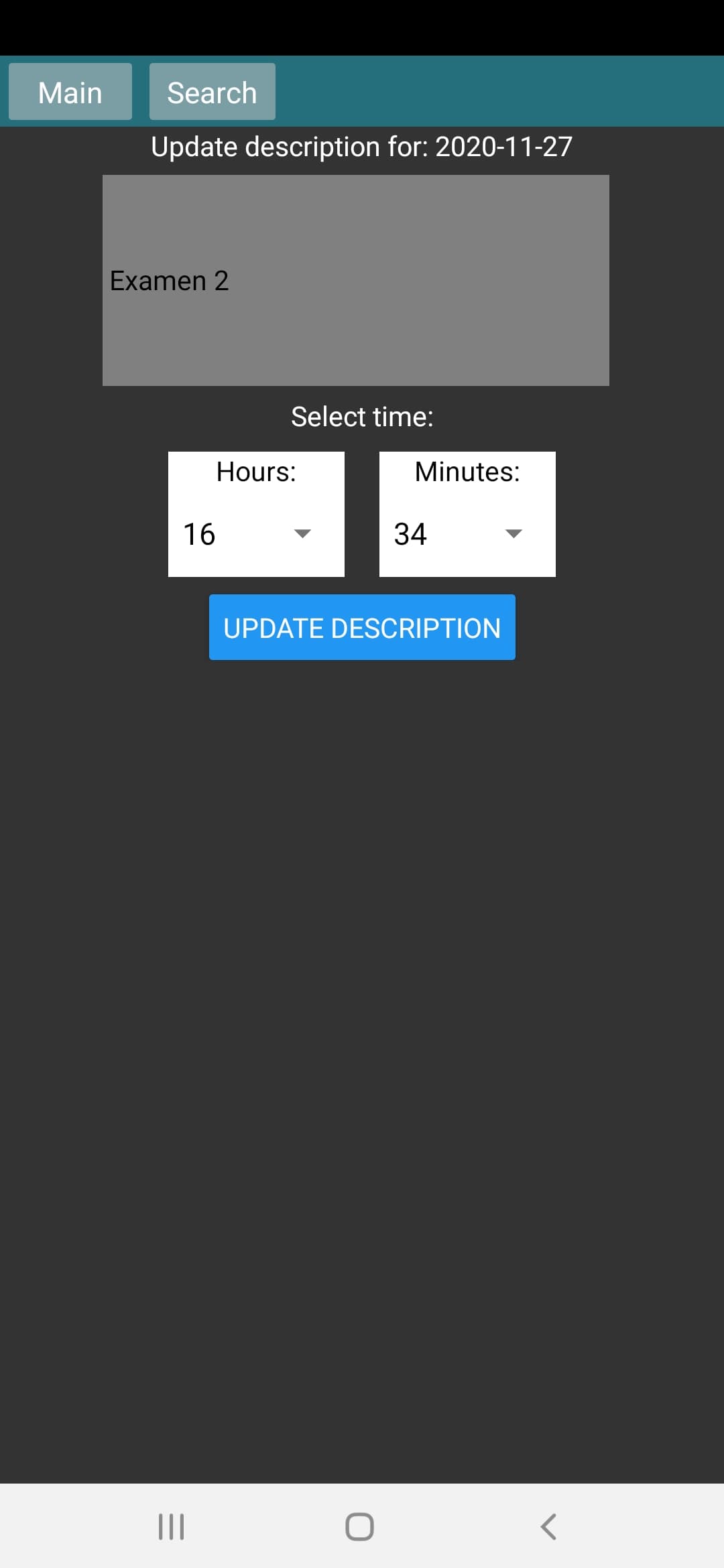
    </Fragment>

  )

}



1. **UpdateActivity** - practic e una și aceeași activitate de la Add, doar ca completata deja. Datele operaționale din interiorul aplicației vor fi stocate în fișier/e XML forma cărora rămâne la discreția personală. (cuvinte cheie, XML Serialization).



 <TextInput

*defaultValue*={this.state.dateTime.description}

*onChangeText*={(text) =>

              this.setState({ dateTime: { ...this.state.dateTime, description: text } })

            }

*multiline*={true}

*style*={styles.inputStyle}

          />

          <TimePicker

*hours*={this.state.dateTime.hours}

*minutes*={this.state.dateTime.minutes}

*onTimeChange*={this.onTimeChange}

          />

          <Button *onPress*={this.onUpdateDescriptionButtonClick} *title*={`Update description`} />