AppDelegate.swift

```
//
//  AppDelegate.swift
//  ProjectManagementTool
//
//  Created by Bror Andreas Nordstrom on 19/05/2019.
//  Copyright © 2019 Bror Andreas Nordstrom. All rights reserved.
//


import UIKit
import CoreData
import IQKeyboardManager
// Qurashi, M. (2018). hackiftekhar/IQKeyboardManager. [online] GitHub. Available at: https://github.com/hackiftekhar/IQKeyboardManager [Accessed 15 May 2019].



@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate, UISplitViewControllerDelegate {

    var window: UIWindow?


    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        // Override point for customization after application launch.

        // IQKeyboardManager is for managing keyboard.
        // Qurashi, M. (2018). hackiftekhar/IQKeyboardManager. [online] GitHub. Available at: https://github.com/hackiftekhar/IQKeyboardManager [Accessed 15 May 2019].

        IQKeyboardManager.shared().isEnabled = true

        let splitViewController = window!.rootViewController as! UISplitViewController
        splitViewController.preferredDisplayMode = .allVisible

//        let navigationController = splitViewController.viewControllers[splitViewController.viewControllers.count-1]
```

```
as! UINavigationController
//        navigationController.topViewController!.navigationItem.leftBarButtonItem
= splitViewController.displayModeButtonItem

        splitViewController.delegate = self
        return true
    }

    func applicationWillResignActive(_ application: UIApplication) {
        // Sent when the application is about to move from active to inactive state.
This can occur for certain types of temporary interruptions (such as an
incoming phone call or SMS message) or when the user quits the application
and it begins the transition to the background state.
        // Use this method to pause ongoing tasks, disable timers, and invalidate
graphics rendering callbacks. Games should use this method to pause the
game.
    }

    func applicationDidEnterBackground(_ application: UIApplication) {
        // Use this method to release shared resources, save user data, invalidate
timers, and store enough application state information to restore your
application to its current state in case it is terminated later.
        // If your application supports background execution, this method is called
instead of applicationWillTerminate: when the user quits.
    }

    func applicationWillEnterForeground(_ application: UIApplication) {
        // Called as part of the transition from the background to the active state;
here you can undo many of the changes made on entering the background.
    }

    func applicationDidBecomeActive(_ application: UIApplication) {
        // Restart any tasks that were paused (or not yet started) while the
application was inactive. If the application was previously in the background,
optionally refresh the user interface.
    }

    func applicationWillTerminate(_ application: UIApplication) {
        // Called when the application is about to terminate. Save data if
appropriate. See also applicationDidEnterBackground:.
    }

    // MARK: - Split view

    func splitViewController(_ splitViewController: UISplitViewController,
collapseSecondary secondaryViewController:UIViewController, onto
```

```
primaryViewController:UIViewController) -> Bool {
        guard let secondaryAsNavController = secondaryViewController as?
UINavigationController else { return false }
        guard let topAsDetailController =
secondaryAsNavController.topViewController as? DetailViewController else
{ return false }
        if topAsDetailController.project == nil {
            // Return true to indicate that we have handled the collapse by doing
nothing; the secondary controller will be discarded.
            return true
        }
        return false
    }

    lazy var persistentContainer: NSPersistentContainer = {

        let container = NSPersistentContainer(name: "DataModel")
        container.loadPersistentStores(completionHandler: { (storeDescription,
error) in
            if let error = error {

                fatalError("Unresolved error, \((error as NSError).userInfo)")
            }
        })
        return container
    }()

}
```

MasterViewController.swift

```
//
//  MasterViewController.swift
//  ProjectManagementTool
//
//  Created by Bror Andreas Nordstrom on 14/05/2019.
//  Copyright © 2019 Bror Andreas Nordstrom. All rights reserved.
//

import UIKit
import CoreData
import EventKit // provides access to calendar

class MasterViewController: UITableViewController {

    var detailViewController: DetailViewController? = nil // child view controller
```

```
    var projects = [Project]() // all projects


    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.

        // get detailViewController
        if let split = splitViewController {
            let controllers = split.viewControllers
            detailViewController = (controllers[controllers.count-1] as!
UINavigationController).topViewController as? DetailViewController
        }

        // load table from Coredata
        self.projects = CoreDataManager._Project.fetch()
        tableView.reloadData()

    }

    override func viewWillAppear(_ animated: Bool) {
        clearsSelectionOnViewWillAppear = splitViewController!.isCollapsed
        super.viewWillAppear(animated)
    }



    @IBAction func onAddProject(_ sender: UIBarButtonItem) {

        // project add button clicked

        // open project add dialog

        let vc = ProjectAddViewController.getInstance() as!
ProjectAddViewController
        vc.preferredContentSize = CGSize(width: 400, height: 550)
        vc.modalPresentationStyle = .popover
        let ppc = vc.popoverPresentationController
        ppc?.permittedArrowDirections = .any
        ppc?.barButtonItem = sender
        present(vc, animated: true, completion: nil)

        vc.delegate = self // self will receive dialog events


    }
```

```
    @IBAction func onEditProject(_ sender: UIBarButtonItem) {

        // project edit button clicked

        // get selected index. otherwise do nothing
        guard let indexPath = self.tableView.indexPathForSelectedRow else
{ return }

        // get selected project
        let project = self.projects[indexPath.row]

        // open project edit dialog
        let vc = ProjectEditViewController.getInstance() as!
ProjectEditViewController

        vc.project = project

        vc.preferredContentSize = CGSize(width: 400, height: 500)
        vc.modalPresentationStyle = .popover
        let ppc = vc.popoverPresentationController
        ppc?.permittedArrowDirections = .any
        ppc?.barButtonItem = sender
        present(vc, animated: true, completion: nil)

        vc.delegate = self // self will receive dialog events


    }

    @IBAction func onAdd2Calendar(_ sender: Any) {

        // add to calendar button clicked

        // get selected index. otherwise do nothing
        guard let indexPath = self.tableView.indexPathForSelectedRow else
{ return }

        // get selected project
        let project = self.projects[indexPath.row]

        // call method to add project to calendar app
        self.addProject2Calendar(project: project)


    }
```

```
    func addProject2Calendar(project: Project, completion: ((_ success: Bool, _
error: NSError?) -> Void)? = nil) {

        // add specific project to calendar

        // use eventstore

        let eventStore = EKEventStore()

        // request access first.
        // this needs Privacy - Calendars Usage Description in info.plist file

        eventStore.requestAccess(to: .event, completion: { (granted, error) in
            if (granted) && (error == nil) {

                // create event for project

                let event = EKEvent(eventStore: eventStore)
                event.title = project.name
                event.startDate = project.startDate! as Date
                event.endDate = project.endDate! as Date
                event.notes = project.note
                event.calendar = eventStore.defaultCalendarForNewEvents
                do {
                    try eventStore.save(event, span: .thisEvent)
                } catch let e as NSError {
                    completion?(false, e)
                    return
                }
                completion?(true, nil)
            } else {
                completion?(false, error as NSError?)
            }
        })

    }

    // MARK: - Segues

    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        if segue.identifier == "showDetail" {
            if let indexPath = tableView.indexPathForSelectedRow {
                let object = projects[indexPath.row]
                let controller = (segue.destination as!
UINavigationController).topViewController as! DetailViewController
```

```
                controller.project = object
//                controller.navigationItem.leftBarButtonItem =
splitViewController?.displayModeButtonItem
//                controller.navigationItem.leftItemsSupplementBackButton = true
            }
        }
    }

    // MARK: - Table View

    override func numberOfSections(in tableView: UITableView) -> Int {
        // this table needs only one section
        return 1
    }

    override func tableView(_ tableView: UITableView, numberOfRowsInSection
section: Int) -> Int {
        // number of rows -> will be same as project count
        return projects.count
    }

    override func tableView(_ tableView: UITableView, cellForRowAt indexPath:
IndexPath) -> UITableViewCell {

        // create cell for project

        let cell = tableView.dequeueReusableCell(withIdentifier:
"MasterTableViewCell", for: indexPath) as! MasterTableViewCell

        // pass project data to cell
        cell.setCellData(project: projects[indexPath.row])

        return cell
    }


}


extension MasterViewController: ProjectAddViewControllerDelegate {

    // on project add dialog, save button is clicked
    func onSave(name: String, note: String, endDate: Date, priority: Int,
add2Calendar: Bool) {

        // save project to coredata
```

```
        let project = CoreDataManager._Project.save(
            name: name,
            note: note,
            endDate: endDate,
            priority: priority)


        if let project = project, add2Calendar {
            // if project is need to be added to calendar app, should call according
method
            self.addProject2Calendar(project: project)
        }

        // fetch projects and reload table
        self.projects = CoreDataManager._Project.fetch()
        tableView.reloadData()

    }

}

extension MasterViewController: ProjectEditViewControllerDelegate {

    // on project edit dialog, save button is clicked
    func onSave(project: Project, name: String, note: String, endDate: Date,
priority: Int) {

        // update selected project on coredata
        CoreDataManager._Project.update(
            project: project,
            name: name,
            note: note,
            endDate: endDate,
            priority: priority)

        // fetch projects and reload table
        self.projects = CoreDataManager._Project.fetch()
        tableView.reloadData()
    }

    // on project edit dialog, delete project is clicked
    func onDelete(project: Project) {

        // delete selected project from coredata
        CoreDataManager._Project.delete(project: project)
```

```
        // fetch projects and reload table
        self.projects = CoreDataManager._Project.fetch()
        tableView.reloadData()

    }

}
```

DetailsViewController.swift

```
//
//  DetailViewController.swift
//  ProjectManagementTool
//
//  Created by Bror Andreas Nordstrom on 14/05/2019.
//  Copyright © 2019 Bror Andreas Nordstrom. All rights reserved.
//


import UIKit
import UserNotifications
import DLLocalNotifications
// Mark: Reference
// Laungani, D. (2018). Devesh Laungani - Welcome. [online] Utdallas. Available at: https://www.utdallas.edu/~dxl141130/ [Accessed 16 May 2019].


class DetailViewController: UIViewController {


    var viewLoadFinished = false // indicates if view is once loaded

    var project: Project? { // project for detail view controller
        didSet {
            // Update the view when project data is set
            configureView()
        }
    }

    @IBOutlet var tableView: UITableView! // task table view

    // below outlet vars are all for project details
    @IBOutlet var labelProjectName: UILabel!
    @IBOutlet var labelProjectNote: UILabel!
    @IBOutlet var pieProjectProgress: PieView!
```

```
    @IBOutlet var labelProjectProgress: UILabel!
    @IBOutlet var pieProjectDayLeft: PieView!
    @IBOutlet var labelProjectDayLeft: UILabel!


    // project details wrapper
    @IBOutlet var stackViewProjectDetails: UIStackView!

    func configureView() {
        // Update the user interface for the detail item.

        if viewLoadFinished == false {
            // if view is not loaded, the UI vars will be nil, this means no need to work
            return
        }

        // reload table data
        self.tableView.reloadData()

        // pie charts must be circular
        self.pieProjectProgress.layer.cornerRadius =
self.pieProjectProgress.frame.width / 2
        self.pieProjectProgress.layer.masksToBounds = true

        self.pieProjectDayLeft.layer.cornerRadius =
self.pieProjectDayLeft.frame.width / 2
        self.pieProjectDayLeft.layer.masksToBounds = true


        // set project details
        if let project = project {

            // if selected project exists, we should show project details
            self.stackViewProjectDetails.isHidden = false

            // show project name
            self.labelProjectName.text = "Project - " + (project.name ?? "")

            // show project note
            self.labelProjectNote.text = project.note

            // if project has tasks, should show task infomation
            if let tasks = project.tasks, tasks.allObjects.count != 0 {

                // holds total task progress sum
                var progressSum: Float = 0
```

```
            // calculate progressSum
            for task in tasks.allObjects {
                progressSum += (task as! Task).progress
            }

            // calculate project progress percentage
            let percent = progressSum / Float(tasks.allObjects.count)

            // set pie chart value
            self.pieProjectProgress.percent = CGFloat(percent)
            //set description label value
            self.labelProjectProgress.text = String(percent) + "% complete"
        } else {
            // if selected project has no tasks, the percent will be 0. set according
data
            self.pieProjectProgress.percent = 0
            self.labelProjectProgress.text = String(0) + "% complete"

        }

        // check if start date and end date are not nil
        if let startDate = project.startDate,
            let endDate = project.endDate {

            // calculate project passed date, progress and days left

            let currentDate = Date()

            // check based on timestamp
            let length = endDate.timeIntervalSince1970 -
startDate.timeIntervalSince1970
            let passed = currentDate.timeIntervalSince1970 -
startDate.timeIntervalSince1970
            let leftDays = Int(((endDate.timeIntervalSince1970 -
currentDate.timeIntervalSince1970) / (3600.0 * 24)).rounded())

            if leftDays < 0  {
                // if leftDays is minus value, the project end date is expired.
                self.pieProjectDayLeft.percent = 100
                self.labelProjectDayLeft.text = String(0) + " days left"
            } else {
                // set project date details data
                let percent = passed * 100.0 / length
                self.pieProjectDayLeft.percent = CGFloat(percent)
                self.labelProjectDayLeft.text = String(leftDays) + " days left"
```

```
            }

        }

    } else {

        // no selected project, don't need to show project details, so, project details wrapper will be hidden
        self.stackViewProjectDetails.isHidden = true

    }

}

override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view, typically from a nib.

    // all table events will be held on self
    self.tableView.delegate = self
    // table data data will be provided from self
    self.tableView.dataSource = self

    // set view loading flag
    viewLoadFinished = true

    // set project and tasks data
    configureView()

    // request authorization for notification
    let center = UNUserNotificationCenter.current()
    // Request permission to display alerts and play sounds.
    center.requestAuthorization(options: [.alert, .sound])
    { (granted, error) in
        // Enable or disable features based on authorization.
    }

}


@IBAction func onAddTask(_ sender: UIBarButtonItem) {

    // task add button is clicked

    // if there is no selected project, just ignore.
```

```
        guard self.project != nil else { return }

        // show task add view controller
        let vc = TaskAddViewController.getInstance() as! TaskAddViewController

        vc.project = project // pass project to dialog

        vc.preferredContentSize = CGSize(width: 400, height: 500)
        vc.modalPresentationStyle = .popover
        let ppc = vc.popoverPresentationController
        ppc?.permittedArrowDirections = .any
        ppc?.barButtonItem = sender
        present(vc, animated: true, completion: nil)

        vc.delegate = self // self will receive dialog events

    }

    @IBAction func onEditTask(_ sender: UIBarButtonItem) {

        // task edit button is clicked

        // ignore if there is no project selected

        guard self.project != nil else { return }

        // ignore if there is no row selected
        guard let indexPath = self.tableView.indexPathForSelectedRow else
{ return }

        // ignore if there is no task selected
        guard let task = self.project?.tasks?.allObjects[indexPath.row] as? Task
else { return }


        // show task edit view controller
        let vc = TaskEditViewController.getInstance() as! TaskEditViewController

        vc.project = project // pass project to dialog
        vc.task = task // pass task to dialog

        vc.preferredContentSize = CGSize(width: 400, height: 500)
        vc.modalPresentationStyle = .popover
        let ppc = vc.popoverPresentationController
        ppc?.permittedArrowDirections = .any
        ppc?.barButtonItem = sender
```

```
        present(vc, animated: true, completion: nil)

        vc.delegate = self // self will receive dialog events

    }

    func addTask2Notification(task: Task) {

        // this methods do work for schedule notification for showing task
information
        // I used DLNotification plugin.
        // Laungani, D. (2018). Devesh Laungani - Welcome. [online] Utdallas.
Available at: https://www.utdallas.edu/~dxl141130/ [Accessed 16 May 2019].

        let triggerDate = task.endDate! as Date

        let notification = DLNotification(
            identifier: "ProjectManagementToolNotificationID" +
                String(Date().timeIntervalSince1970),
            alertTitle: task.name ?? "",
            alertBody: task.note ?? "",
            date: triggerDate,
            repeats: .none)

        let scheduler = DLNotificationScheduler()
        scheduler.scheduleNotification(notification: notification)
        scheduler.scheduleAllNotifications()



    }

}

extension DetailViewController: UITableViewDelegate {

}

extension DetailViewController: UITableViewDataSource {
    func tableView(_ tableView: UITableView, numberOfRowsInSection section:
Int) -> Int {

        // return number of rows. if there is no project selected, should return 0
        return self.project?.tasks?.count ?? 0
    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath)
```

```
-> UITableViewCell {

        // create cell for each task
        let cell = tableView.dequeueReusableCell(withIdentifier:
"DetailTableViewCell", for: indexPath) as! DetailTableViewCell

        // set task data to cell
        if let object = self.project?.tasks?.allObjects[indexPath.row],
           let project = self.project {

           // pass data
           cell.setCellData(project: project, task: object as! Task)

        }


        return cell
    }


}

extension DetailViewController: TaskAddViewControllerDelegate {

    // on task add dialog, save button is clicked

    func onSave(name: String, note: String, endDate: Date, addNotification:
Bool) {

        // check if selected project exists
        guard let project = self.project else { return }

        // save task to coredata
        let task = CoreDataManager._Task.save(
           project: project,
           name: name,
           note: note,
           endDate: endDate)

        // if needs to add notification, do work for it
        if let task = task, addNotification {

           // call method for add to notification
           self.addTask2Notification(task: task)

        }
```

```
        // update view

        self.configureView()


    }


}

extension DetailViewController: TaskEditViewControllerDelegate {


    // on task edit dialog, save button is clicked
    func onSave(project: Project, task: Task, name: String, note: String, endDate:
Date, progress: Float) {

        // update task for project on coredata
        CoreDataManager._Task.update(
            project: project,
            task: task,
            name: name,
            note: note,
            endDate: endDate,
            progress: progress)

        // update view
        self.configureView()
    }

    // on task edit dialog, delete button is clicked
    func onDelete(project: Project, task: Task) {

        // delete task from coredata
        CoreDataManager._Task.delete(
            project: project,
            task: task)

        // update view
        self.configureView()

    }

}
```

ProjectAddViewController.swift

```
//
//  ProjectAddViewController.swift
//  ProjectManagementTool
//
//  Created by Bror Andreas Nordstrom on 15/05/2019.
//  Copyright © 2019 Bror Andreas Nordstrom. All rights reserved.
//


import Foundation
import UIKit

// protocol for project add dialog events
protocol ProjectAddViewControllerDelegate {
    // this is for save button
    func onSave(name: String, note: String, endDate: Date, priority: Int,
add2Calendar: Bool)
}

// project add dialog
class ProjectAddViewController: UIViewController {

    var delegate: ProjectAddViewControllerDelegate? // delegate for event


    // outlet vars for UI
    @IBOutlet var textFieldProjectName: UITextField!
    @IBOutlet var textFieldNotes: UITextField!
    @IBOutlet weak var datePickerEndDate: UIDatePicker!
    @IBOutlet weak var segmentPriority: UISegmentedControl!
    @IBOutlet weak var switchAdd2Calendar: UISwitch!

    @IBAction func onSave(_ sender: Any) {

        // save button is clicked

        // validate inputs
        if (textFieldProjectName.text ?? "") == "" {
            return
        }

        if (textFieldNotes.text ?? "") == "" {
            return
```

```
        }

        // all inputs are valid

        // close dialog
        self.dismiss(animated: true) {

            // after dialog is closed, should emit event via delegate object
            self.delegate?.onSave(
                name: self.textFieldProjectName.text ?? "",
                note: self.textFieldNotes.text ?? "",
                endDate: self.datePickerEndDate.date,
                priority: self.segmentPriority.selectedSegmentIndex,
                add2Calendar: self.switchAdd2Calendar.isOn)
        }

    }

    override func viewDidLoad() {
        super.viewDidLoad()

        // set minimum date. the project end date min date will be today
        self.datePickerEndDate.minimumDate = Date()

    }

    static func getInstance() -> UIViewController {

        // create dialog instance from storyboard

        let vc = UIStoryboard(name: "Main", bundle:
nil).instantiateViewController(withIdentifier: "ProjectAddViewController") as
UIViewController

        return vc
    }

}
```

ProjectEditViewController.swift

```
//
//  ProjectEditViewController.swift
//  ProjectManagementTool
//
//  Created by Bror Andreas Nordstrom on 15/05/2019.
```

```
//  Copyright © 2019 Bror Andreas Nordstrom. All rights reserved.
//


import Foundation
import UIKit

// protocol for project edit dialog events
protocol ProjectEditViewControllerDelegate {

    // for save button
    func onSave(project: Project, name: String, note: String, endDate: Date,
priority: Int)

    // for delete button
    func onDelete(project: Project)
}

// projecte deit dialog
class ProjectEditViewController: UIViewController {

    var delegate: ProjectEditViewControllerDelegate? // delegate for event

    var project: Project? // selected project. passed from parent

    // outlet vars for UI
    @IBOutlet var textFieldProjectName: UITextField!
    @IBOutlet var textFieldNotes: UITextField!
    @IBOutlet weak var datePickerEndDate: UIDatePicker!
    @IBOutlet weak var segmentPriority: UISegmentedControl!

    @IBAction func onSave(_ sender: Any) {

        // save button clicked

        // validate inputs
        if (textFieldProjectName.text ?? "") == "" {
            return
        }

        if (textFieldNotes.text ?? "") == "" {
            return
        }

        // all inputs are valid
```

```
        // close dialog

        self.dismiss(animated: true) {

            // after dialog is closed, emit event via delegate object
            self.delegate?.onSave(
                project: self.project!,
                name: self.textFieldProjectName.text ?? "",
                note: self.textFieldNotes.text ?? "",
                endDate: self.datePickerEndDate.date,
                priority: self.segmentPriority.selectedSegmentIndex)
        }

    }

    @IBAction func onDelete(_ sender: Any) {

        // delete button cliekced

        // close dialog
        self.dismiss(animated: true) {

            // after dialog is closed, emit event via delegate object
            self.delegate?.onDelete(project: self.project!)
        }


    }

    // show project data to dialog
    func configureView() {

        // check if project is not nil
        guard let project = project else { return }


        // show data to UI
        self.textFieldProjectName.text = project.name
        self.textFieldNotes.text = project.note
        self.datePickerEndDate.setDate(project.endDate! as Date, animated:
false)
        self.segmentPriority.selectedSegmentIndex = Int(project.priority)

    }

    override func viewDidLoad() {
```

```
        super.viewDidLoad()

        // project end date min value will be today
        self.datePickerEndDate.minimumDate = Date()

        // show data to UI
        self.configureView()

    }

    static func getInstance() -> UIViewController {

        // create instance for dialog from storyboard
        let vc = UIStoryboard(name: "Main", bundle:
nil).instantiateViewController(withIdentifier: "ProjectEditViewController") as
UIViewController

        return vc
    }

}
```

TaskAddViewController.swift

```
//
//  TaskAddViewController.swift
//  ProjectManagementTool
//
//  Created by Bror Andreas Nordstrom on 17/05/2019.
//  Copyright © 2019 Bror Andreas Nordstrom. All rights reserved.
//


import Foundation
import UIKit

// protocol for task add dialog events
protocol TaskAddViewControllerDelegate {

    // for save button
    func onSave(name: String, note: String, endDate: Date, addNotification:
Bool)
}

// task add dialog
```

```
class TaskAddViewController: UIViewController {

    var delegate: TaskAddViewControllerDelegate? // delegate object for events
    var project: Project? // project for new task. passed from parent

    // outlet vars for UI
    @IBOutlet var textFieldName: UITextField!
    @IBOutlet var textFieldNotes: UITextField!
    @IBOutlet weak var datePickerEndDate: UIDatePicker!
    @IBOutlet weak var switchAddNotification: UISwitch!

    @IBAction func onSave(_ sender: Any) {

        // save button is clicked

        // should validate all inputs
        if (textFieldName.text ?? "") == "" {
            return
        }

        if (textFieldNotes.text ?? "") == "" {
            return
        }

        // all inputs are valid

        // close dialog
        self.dismiss(animated: true) {

            // after dialog is closed, emit event via delegate object
            self.delegate?.onSave(
                name: self.textFieldName.text ?? "",
                note: self.textFieldNotes.text ?? "",
                endDate: self.datePickerEndDate.date,
                addNotification: self.switchAddNotification.isOn)

        }

    }

    override func viewDidLoad() {
        super.viewDidLoad()

        // set date picker min and max value. min = today, max = project end date
        self.datePickerEndDate.minimumDate = Date()
```

```
        self.datePickerEndDate.maximumDate = self.project?.endDate as Date?

    }

    static func getInstance() -> UIViewController {

        // create instance for dialog from storybaord
        let vc = UIStoryboard(name: "Main", bundle:
nil).instantiateViewController(withIdentifier: "TaskAddViewController") as
UIViewController

        return vc
    }



}
```

TaskEditViewController.swift

```
//
//  TaskEditViewController.swift
//  ProjectManagementTool
//
//  Created by Bror Andreas Nordstrom on 17/05/2019.
//  Copyright © 2019 Bror Andreas Nordstrom. All rights reserved.
//


import Foundation
import UIKit

// protocol for task edit dialog events
protocol TaskEditViewControllerDelegate {

    // for save button
    func onSave(project: Project, task: Task, name: String, note: String, endDate:
Date, progress: Float)

    // for delete button
    func onDelete(project: Project, task: Task)

}
```

```swift
// task edit dialog
class TaskEditViewController: UIViewController {

    var delegate: TaskEditViewControllerDelegate? // delegate object for event
    var project: Project? // project for task. passed from parent
    var task: Task? // task to be edited. passed from parent


    // outlet vars for UI
    @IBOutlet var textFieldName: UITextField!
    @IBOutlet var textFieldNotes: UITextField!
    @IBOutlet var datePickerEndDate: UIDatePicker!
    @IBOutlet var sliderProgress: UISlider!

    @IBAction func onSave(_ sender: Any) {

        // save button clicked


        // need to validate inputs

        if (textFieldName.text ?? "") == "" {
            return
        }

        if (textFieldNotes.text ?? "") == "" {
            return
        }

        // all inputs are valid

        // close dialog
        self.dismiss(animated: true) {


            // after dialog is closed, we should emit event via delegate object
            self.delegate?.onSave(
                project: self.project!,
                task: self.task!,
                name: self.textFieldName.text ?? "",
                note: self.textFieldNotes.text ?? "",
                endDate: self.datePickerEndDate.date,
                progress: (self.sliderProgress.value * 10).rounded() / 10
            )

        }
```

```
    }


    @IBAction func onDelete(_ sender: Any) {

        // delete button is clicked

        // close dialog
        self.dismiss(animated: true) {

            // emit event via delegate object
            self.delegate?.onDelete(
                project: self.project!,
                task: self.task!)
        }

    }


    func configureView() {

        // show data on UI

        self.textFieldName.text = self.task?.name
        self.textFieldNotes.text = self.task?.note
        self.datePickerEndDate.setDate(self.task!.endDate! as Date, animated:
false)
        self.sliderProgress.value = self.task?.progress ?? 0

    }

    override func viewDidLoad() {
        super.viewDidLoad()

        // set date picker min and max value. min = today, max = project end date
        self.datePickerEndDate.minimumDate = Date()

        self.datePickerEndDate.maximumDate = self.project?.endDate as Date?

        // show data to UI
        self.configureView()

    }

    static func getInstance() -> UIViewController {
```

```
        // create dialog instance from storyboard

        let vc = UIStoryboard(name: "Main", bundle:
nil).instantiateViewController(withIdentifier: "TaskEditViewController") as
UIViewController

        return vc
    }

}
```

MasterTableViewCell.swift

```
//
//  MasterTableViewCell.swift
//  ProjectManagementTool
//
//  Created by Bror Andreas Nordstrom on 17/05/2019.
//  Copyright © 2019 Bror Andreas Nordstrom. All rights reserved.
//


import Foundation
import UIKit

// class for project table view cell
class MasterTableViewCell: UITableViewCell {

    // outlet vars for UI
    @IBOutlet var labelName: UILabel!
    @IBOutlet var labelEndDate: UILabel!
    @IBOutlet var labelNote: UILabel!
    @IBOutlet weak var priorityView: UIView!



    func setCellData(project: Project) { // set cell data from paretn

        // project name
        self.labelName.text = project.name

        // project date info
        if let startDate = project.startDate,
            let endDate = project.endDate {
```

```
            self.labelEndDate.text = startDate.toString() + " - " + endDate.toString()
        }

        // project note
        self.labelNote.text = project.note

        // project priority
        let colors: [Int32: UIColor] = [
            0: UIColor.red,
            1: UIColor.green,
            2: UIColor.blue
        ]

        self.priorityView.backgroundColor = colors[project.priority]

    }

}
```

DetailTableViewCell.swift

```
//
//  DetailTableViewCell.swift
//  ProjectManagementTool
//
//  Created by Bror Andreas Nordstrom on 17/05/2019.
//  Copyright © 2019 Bror Andreas Nordstrom. All rights reserved.
//


import Foundation
import UIKit

// class for task table view cell
class DetailTableViewCell: UITableViewCell {

    // outlet vars for UI
    @IBOutlet var labelName: UILabel!
    @IBOutlet var labelDate: UILabel!
    @IBOutlet var labelNote: UILabel!
    @IBOutlet var labelPercent: UILabel!
    @IBOutlet var progressView: ProgressView!
    @IBOutlet weak var pieView: PieView!

    override func layoutSubviews() {
        super.layoutSubviews()
```

```
        // we should set pie view circular
        self.pieView.layer.cornerRadius = self.pieView.frame.width / 2
        self.pieView.layer.masksToBounds = true

    }


    func setCellData(project: Project, task: Task) { // set cell data from parent

        // task name
        self.labelName.text = task.name

        // task note
        self.labelNote.text = task.note

        // set task date info
        if let startDate = task.startDate,
            let endDate = task.endDate {

            self.labelDate.text = startDate.toString() + " - " + endDate.toString()

            // calcuate length and percent for date based on timestamp
            let currentDate = Date()
            let length = endDate.timeIntervalSince1970 -
startDate.timeIntervalSince1970
            let passed = currentDate.timeIntervalSince1970 -
startDate.timeIntervalSince1970

            let percent = 100.0 * passed / length

            // set progress
            self.progressView.percent = CGFloat(percent)
        }

        // set pie view info
        self.setPiePercent(CGFloat(task.progress))

    }

    func setPiePercent(_ percent: CGFloat) {
        // set pie view data
        self.pieView.percent = percent
        // set pie view description label data
        self.labelPercent.text = "" + String(Float(percent)) + "%"
    }
```

}

ProgressView.swift

```
//
//  ProgressView.swift
//  ProjectManagementTool
//
//  Created by Bror Andreas Nordstrom on 17/05/2019.
//  Copyright © 2019 Bror Andreas Nordstrom. All rights reserved.
//


import Foundation
import UIKit

// simple progress view
class ProgressView: UIView {

    var percent: CGFloat? {
        didSet {
            // re-draw view when percent data is set
            self.setNeedsDisplay()
        }
    }

    override func draw(_ rect: CGRect) {

        // draw

        // get context
        guard let ctx = UIGraphicsGetCurrentContext() else { return }

        guard let percent = self.percent else { return }

        let width = self.frame.width * percent / 100.0

        // draw two rectangles

        // background rectangle
        ctx.setFillColor(UIColor.darkGray.cgColor)
        ctx.fill(CGRect(x: 0, y: 0, width: self.frame.width, height:
self.frame.height))

        // percent rectangle
```

```
        ctx.setFillColor(self.tintColor.cgColor)
        ctx.fill(CGRect(x: 0, y: 0, width: width, height: self.frame.height))
    }

}
```

PieView.swift

```
//
//  PieView.swift
//  ProjectManagementTool
//
//  Created by Bror Andreas Nordstrom on 17/05/2019.
//  Copyright © 2019 Bror Andreas Nordstrom. All rights reserved.
//


import Foundation
import UIKit

// simple pie view
class PieView: UIView {

    var percent: CGFloat? {
        didSet {

            // re-draw when percent data is set
            self.setNeedsDisplay()
        }
    }



    override func draw(_ rect: CGRect) {

        // draw

        // get context for drawing
        guard let ctx = UIGraphicsGetCurrentContext() else { return }

        guard let percent = self.percent else { return }

        // draw rectangle for background
        ctx.setFillColor(UIColor.darkGray.cgColor)
        ctx.fill(CGRect(x: 0, y: 0, width: self.frame.width, height:
```

```
self.frame.height))

        // calculate pie radius
        let radius = self.frame.width / 2

        // get path for pie
        let path = UIBezierPath.init(circleSegmentCenter: CGPoint(x: radius, y:
radius), radius: radius, startAngle: 270 - ( 360.0 * percent / 100.0 ) , endAngle:
270)

        // set fill color
        ctx.setFillColor(self.tintColor.cgColor)

        // fill path
        path.fill()

        ctx.addPath(path.cgPath)

        ctx.fillPath()


    }


}
```

Extensions.swift

```
//
//  Extensions.swift
//  ProjectManagementTool
//
//  Created by Bror Andreas Nordstrom on 17/05/2019.
//  Copyright © 2019 Bror Andreas Nordstrom. All rights reserved.
//


import Foundation
import UIKit

// extension for NSDate
extension NSDate {

    // returns string represents date
    func toString(_ format: String = "yyyy/MM/dd") -> String {
        /*
```

```
        yyyy.MM.dd
        yyyy-MM-dd HH:mm:ss
        yyyy/MM/dd
        */
       // use date formatter
       let dateFormatter = DateFormatter()
       dateFormatter.dateFormat = format
       return dateFormatter.string(from: self as Date)
   }

}

// extension for CGFlaot
extension CGFloat {
   // returns radians value from angle value
   func radians() -> CGFloat {
      let b = CGFloat(Double.pi) * (self/180)
      return b
   }
}

// extension for UIBezierPath
extension UIBezierPath {

   // create constructor for generating path from pie data
   convenience init(circleSegmentCenter center:CGPoint, radius:CGFloat,
startAngle:CGFloat, endAngle:CGFloat)
   {
      self.init()

      // move to center
      self.move(to: CGPoint.init(x: center.x, y: center.y))

      // draw arc for pie
      self.addArc(withCenter: center, radius: radius, startAngle:
startAngle.radians(), endAngle: endAngle.radians(), clockwise:true)

      // close path
      self.close()
   }
}
```

CoreDataManager.swift

```
//
```

```swift
//  CoreDataManager.swift
//  ProjectManagementTool
//
//  Created by Bror Andreas Nordstrom on 17/05/2019.
//  Copyright © 2019 Bror Andreas Nordstrom. All rights reserved.
//


import Foundation
import UIKit
import CoreData

// class for managing core data
class CoreDataManager {

    // class for managing projects
    public class _Project {

        // returns all projects
        static func fetch() -> [Project] {

            // get app delegate
            guard let appDelegate = UIApplication.shared.delegate as?
AppDelegate else { return [] }

            // get persistentContainer context
            let managedContext = appDelegate.persistentContainer.viewContext

            // prepare fetch
            let projectFetch =
NSFetchRequest<NSFetchRequestResult>(entityName: "Project")

            // fetch projects
            let projects = try! managedContext.fetch(projectFetch)

            return projects as! [Project]

        }

        static func save(name: String, note: String, endDate: Date, priority: Int) ->
Project? {

            // save new project

            // get app delegate
            guard let appDelegate = UIApplication.shared.delegate as?
```

```
AppDelegate else { return nil }

        // get persistentContainer context
        let managedContext = appDelegate.persistentContainer.viewContext

        // prepare entity
        let projectEntity = NSEntityDescription.entity(forEntityName: "Project",
in: managedContext)!

        // create new project object
        let project = NSManagedObject(entity: projectEntity, insertInto:
managedContext)

        // set data for project
        project.setValue(name, forKeyPath: "name")
        project.setValue(note, forKey: "note")
        project.setValue(endDate, forKey: "endDate")
        project.setValue(priority, forKey: "priority")
        project.setValue(Date(), forKey: "startDate")


        // save context

        do {
           try managedContext.save()
        } catch let error as NSError {
           print("Could not save. \(error), \(error.userInfo)")
        }

        return project as? Project

    }

    static func update(project: Project, name: String, note: String, endDate:
Date, priority: Int) {

        // update project data on core data

        // get app delegate
        guard let appDelegate = UIApplication.shared.delegate as?
AppDelegate else { return }

        // get persistentContainer context
        let managedContext = appDelegate.persistentContainer.viewContext

        // set data for project
```

```
            project.setValue(name, forKeyPath: "name")
            project.setValue(note, forKey: "note")
            project.setValue(endDate, forKey: "endDate")
            project.setValue(priority, forKey: "priority")
            project.setValue(Date(), forKey: "startDate")

            // save context

            do {
                try managedContext.save()
            } catch let error as NSError {
                print("Could not save. \(error), \(error.userInfo)")
            }

        }

        static func delete(project: Project) {

            // delete project from core data

            // get app delegate
            guard let appDelegate = UIApplication.shared.delegate as?
AppDelegate else { return }

            // get persistentContainer context
            let managedContext = appDelegate.persistentContainer.viewContext

            // delete all tasks for project
            for task in project.tasks ?? [] {
                managedContext.delete(task as! Task)
            }

            // delete project
            managedContext.delete(project)

            // save context

            do {
                try managedContext.save()
            } catch let error as NSError {
                print("Could not save. \(error), \(error.userInfo)")
            }
        }
    }

    public class _Task { // class for managing projects
```

```
    static func fetch() -> [Task] {

        // get tasks stored on core data

        // get app delegate
        guard let appDelegate = UIApplication.shared.delegate as?
AppDelegate else { return [] }

        // get persistentContainer context
        let managedContext = appDelegate.persistentContainer.viewContext

        // prepare fetch
        let taskFetch = NSFetchRequest<NSFetchRequestResult>(entityName:
"Task")

        // fetch tasks
        let tasks = try! managedContext.fetch(taskFetch)

        return tasks as! [Task]

    }

    static func save(project: Project, name: String, note: String, endDate:
Date) -> Task? {

        // save new task to core data

        // get app delegate
        guard let appDelegate = UIApplication.shared.delegate as?
AppDelegate else { return nil }

        // get persistentContainer context
        let managedContext = appDelegate.persistentContainer.viewContext

        // prepare entity
        let taskEntity = NSEntityDescription.entity(forEntityName: "Task", in:
managedContext)!

        // create task object
        let task = NSManagedObject(entity: taskEntity, insertInto:
managedContext)

        // set data for task
        task.setValue(name, forKeyPath: "name")
        task.setValue(note, forKey: "note")
        task.setValue(endDate, forKey: "endDate")
```

```
        task.setValue(0, forKey: "progress")
        task.setValue(project, forKey: "project")
        task.setValue(Date(), forKey: "startDate")

        // save context

        do {
            try managedContext.save()
        } catch let error as NSError {
            print("Could not save. \(error), \(error.userInfo)")
        }

        return task as? Task

    }

    static func update(project: Project, task: Task, name: String, note: String,
endDate: Date, progress: Float) {

        // update task on core data

        // get app delegate
        guard let appDelegate = UIApplication.shared.delegate as?
AppDelegate else { return }

        // get persistentContainer context
        let managedContext = appDelegate.persistentContainer.viewContext

        // set data for task
        task.setValue(name, forKeyPath: "name")
        task.setValue(note, forKey: "note")
        task.setValue(endDate, forKey: "endDate")
        task.setValue(progress, forKey: "progress")
        task.setValue(project, forKey: "project")
        task.setValue(Date(), forKey: "startDate")

        // save context

        do {
            try managedContext.save()
        } catch let error as NSError {
            print("Could not save. \(error), \(error.userInfo)")
        }

    }
```

```
        static func delete(project: Project, task: Task) {
            // delete task from project and core data

            // get app delegate
            guard let appDelegate = UIApplication.shared.delegate as?
AppDelegate else { return }

            // get persistentContainer context
            let managedContext = appDelegate.persistentContainer.viewContext

            // remove task from project
            project.removeFromTasks(task)

            // delete task
            managedContext.delete(task)

            // save context
            do {
                try managedContext.save()
            } catch let error as NSError {
                print("Could not save. \(error), \(error.userInfo)")
            }
        }

    }


}
```