math_2020_032

Міністерство освіти і науки України

Чернівецький національний університет

імені Юрія Федьковича

Факультет математики та інформатики
(повна назва інституту/факультету)

Кафедра математичного моделювання
(повна назва кафедри)

Додаток під операційну систему iOS "HR Hub"

Дипломна робота

Рівень вищої освіти - другий (магістерський)

Виконав (ла):
студент (ка) 6 курсу, групи 607
спеціальності 124 – Системний аналіз_
(назва спеціальності)
____Городенський Павло Андрійович_
(прізвище, ім'я та по-батькові)
Керівник а.ф.-м.н., доцент Перцов А.С.
(науковий ступінь, вчене звання, прізвище та ініціали)

До захисту допущено:

Протокол засідання кафедри № 6

від „3" грудня 2019 р.

зав. кафедри _____ доц. Піддубна Л.А.

Чернівці – 2019

АНОТАЦІЯ

В курсовій роботі розглядається процес створення IOS - додатку на мові програмування «Swift 5» для полегшення контролю людськими ресурсами в рінхих компаніях. Додаток був написаний за допомогою фреймворків «Alamofire.framework», «UIKit.framework» , «SDWebImage.framework», «Photos.framework», «Fondation.framework», «SVProgressHUD.framework» тощо.

ЗМІСТ

ВСТУП

Серед управління діяльністю підприємства, найскладнішим є менеджмент персоналу. Саме персонал являє собою рушійну силу підприємства, як соціально-економічну й виробничу системи, що є найважливішим фактором його розвитку на сучасному етапі ринкових відносин [1]. Дослідження з управління людськими ресурсами вже протягом кількох десятиліть зосередженні на вивченні механізмів впливу HR на ефективну працездатність персоналу. Наразі на підприємствах використовуються більш складні методики аналітичної діяльності, у тому числі й HR - аналітика, що може значно посилити результативність процесів прийняття управлінських рішень й уникнути помилок.

Розвиток ринкової економіки створює умови, при яких зростає значення людського фактору в робочій організації. Знання, кваліфікація, досвід, творчий потенціал працівників є нематеріальним активом компанії. Вони стають одним із головних факторів її ефективності та конкурентоспроможності [2].

Політика управління по різному поширена й розвинута по всьому світу. Сучасні тренди у сфері аналітики управління персоналом підприємницької діяльності закордоном мають свої переваги над українськими, тому потребують детальнішого дослідження.

На зміну уявлення про кадрову роботу прийшло - управління персоналом, як особливий напрям в системі штабної діяльності. При цьому виявляється, що крім обліку необхідно створити і оптимізувати внутрішні організаційні соціальні процеси, тобто сформувати кадровий потенціал і виробити заходи з розвитку персоналу відповідно до цілей і принципів корпоративної політики. Іншими словами необхідно зіставити економічні, технологічні, інформаційні, структурні процеси в організації і поза нею з процесами.

Запропонований підхід інтеграції HR для мобільного додатку під операційну систему iOS, основна задача якого є вплив на об’єкт управління з метою реалізації управлінського рішення.

Іншими словами, додаток призначений для оптимізації та автоматизації трьох традиційних стовпів управління персоналом: заробітної плати, часу і відвідуваності, а також управління перевагами.

Робота складається з двох частин. У першій частині розглядаються основні поняття технологій управління та їх основні напрямки. Друга частина роботи є практичною, в якій продемонстровано роботу програмного забезпечення під операційну систему iOS.

РОЗДІЛ 1

ІСТОРІЯ ТА РОЗВИТОК НАПРЯМУ HR МЕНЕДЖМЕНТУ

1.1 Історія професії HR-менеджер

Поле людських ресурсів почало формуватися в Європі 18 століття. Він побудований на простій ідеї Роберта Оуена (1771-1858) та Чарльза Беббіджа (1791-1871) під час промислової революції. Ці люди зробили висновок, що люди мають вирішальне значення для успіху організації. Вони висловлювали думку, що добробут працівників приводить до бездоганної роботи, без здорових робітників організація не виживе [3].

HR як специфічна сфера, виникла на початку 20 століття, під впливом Фредеріка Вінслоу Тейлора (1856–1915). Тейлор досліджував те, що він називав "науковим менеджментом" (іноді його називали "тейлоризмом"), прагнучи підвищити економічну ефективність на виробництві. Врешті решт він зосередився на одному з головних вкладів у виробничий процес праці - іскровому дослідженні, щодо продуктивності робочої сили [4].

Тим часом в Англії С. Майєрс, натхненний несподіваними проблемами серед солдатів, які стривожили полководців та політиків у Першій світовій війні 1914-1918 років, створив у 1921 році Національний інститут промислової психології (NIIP) [5]. Роблячи це, він створив насіння для руху людських відносин. Цей рух, по обидва боки Атлантики, побудований на дослідженнях Елтона Майо (1880-1949) та інших для документування за допомогою досліджень Готорна (1924–1932) та інших досліджень, як стимули, не пов'язані з фінансовою компенсацією та умовами праці, можуть врожайніші продуктивніші працівники. [6] Робота Абрахама Маслоу (1908–1970), Курта Левіна (1890–1947), Макса Вебера (1864–1920), Фредеріка Герцберга (1923–2000) та Девіда МакКлелленда (1917–1998), що є основою для досліджень у галузі промисловості та організаційна психологія, організаційна поведінка та організаційна теорія тлумачились таким чином, щоб подальше заявляти про законність застосовуваної дисципліни.

1.2 Народження та розвиток дисципліни

На той час існувало достатньо теоретичних доказів, щоб зробити бізнес-приклад для стратегічного управління робочою силою, змін у бізнес-ландшафт та державній політиці. Угода перетворила відносини роботодавець-працівник, і дисципліна стала формалізована як "виробничі та трудові відносини". У 1913 р. Одна з найстаріших відомих професійних асоціацій з персоналу - Чартерний інститут персоналу та розвитку - заснована в Англії як Асоціація робітників добробуту, вона змінила назву через десять років на Інститут працівників промислового добробуту, і знову наступне десятиліття - на Інститут управління праці, перш ніж визначитися з теперішньою назвою в 2000 році [7]. Так само у Сполучених Штатах в університеті Корнелла у 1945 р. Сформувався перший у світі вищий навчальний заклад, присвячений навчанню на робочому місці - Школа виробничих і трудових відносин [8]. У 1948 р. те, що згодом перетвориться на найбільшу професійну асоціацію з управління персоналом - Товариство управління людськими ресурсами (SHRM) - утворилося як Американське товариство управління персоналом [9].

Тим часом у Радянському Союзі сталінське використання меценатства, здійснюваного через еквівалент "відділу кадрів" у більшовицькій партії продемонструвало ефективність та вплив кадрової політики та практики [10], і сам Сталін визнав важливість людського ресурсу, наприклад, у його масовому розміщенні його в системі ГУЛАГ.

Протягом останньої половини XX століття членство в профспілці значно скоротилося, тоді як управління робочою силою продовжувало розширювати свій вплив у межах організацій. У США словосполучення "виробничі та трудові відносини" вживається для конкретного посилання на питання, що стосуються колективне представництво, і багато компаній почали називати професію "адміністрацією кадрів". Багато сучасних практик управління персоналом виникли з потребою компаній у 1950-х роках для розвитку та збереження талантів [11].

В кінці 20 століття прогрес у галузі транспорту та зв'язку значно полегшив мобільність робочої сили та співпрацю. Корпорації почали розглядати працівників як активи. Отже, "управління людськими ресурсами" стало домінуючим терміном функції, навіть змінила назву на Товариство управління людськими ресурсами у 1998 році [12].

"Управління людським капіталом" іноді використовується синонімом до "HR", хоча "людський капітал" зазвичай стосується більш вузького погляду на людські ресурси; тобто знання, які люди втілюють і можуть сприяти організації. Так само інші терміни, які іноді використовуються для опису галузі, включають "організаційний менеджмент", "управління персоналом", "управління талантами", "управління персоналом" та просто "управління людьми".

1.3 Функція бізнесу HR

Дейв Ульріх перераховує функції HR: узгодження HR та бізнес-стратегії, перепрофілювання організаційних процесів, прослуховування та реагування на співробітників, а також управління трансформацією та змінами [13].

На макрорівні HR відповідає за нагляд за організаційним лідерством та культурою. HR також забезпечує дотримання законодавства про зайнятість та працю, які відрізняються між собою за географією та часто наглядають за охороною здоров'я, безпекою та безпекою. Виходячи з географічного положення, існують різні закони. Існує кілька федеральних законів, які мають вирішальне значення для ознайомлення менеджерів з персоналу, щоб захистити як свою компанію, так і її працівників. Важливі федеральні закони та нормативні акти включають Закон про норми справедливої праці, який передбачає встановлення мінімальної заробітної плати та захист права певних працівників на заробіток. Закон про відпустку у справах сім'ї та лікарняних працівників надає право на отримання працівникам до 12 тижнів неоплаченої відпустки з сімейних та медичних причин. Забезпечення відповідності компанії всім законам та нормам є важливим аспектом роботи з персоналом та захищатиме компанію від будь-якої «юридичної відповідальності» [14]. За обставин, коли працівники бажають та

мають юридичну правомочність укладати колективний договір, HR зазвичай також виконує функції первинного зв'язку компанії з представниками працівника (як правило, профспілки). Отже, HR, як правило, через представників, бере участь у лобіювання зусиль з урядовими установами.

Управління людськими ресурсами має чотири основні функції: персонал, навчання та розвиток, мотивація та обслуговування. Штатний персонал - це підбір потенційних працівників, які проводяться шляхом співбесіди, заявок, встановлення мереж тощо. До складу персоналу є два основні фактори: залучення талановитих рекрутів та найм ресурсів. Менеджери з персоналу повинні створити детальні стратегії підбору персоналу та мати план дій, який слід висувати при підборі персоналу. Далі менеджери можуть застосовувати стратегії за допомогою найму ресурсів, розширюючи, щоб знайти найкращих можливих рекрутів для команди.

Рекрутинг є дуже конкурентоспроможним, оскільки кожна компанія хоче найкращих кандидатів [15]. Використання таких тактик, як засоби масової інформації, може привернути увагу потенційних рекрутів [16]. Навчання та розвиток - наступний крок і включає постійний процес навчання та розвиток компетентних та адаптованих працівників. Тут мотивація розглядається як ключова для того, щоб підтримувати працівників високопродуктивними. Сюди входять виплати працівникам, оцінки ефективності роботи та винагороди. Пільги працівникам, оцінювання та винагорода - все це заохочення для залучення кращих працівників. Остання функція, обслуговування, полягає у збереженні відданості працівників та лояльності до організації. Деякі компанії глобалізуються та формують більш різноманітні команди. Відділи кадрів мають роль гарантувати, що ці команди можуть функціонувати та що люди можуть спілкуватися через культуру та через кордон. Дисципліна може також брати участь в управлінні мобільністю, особливо для емігрантів, і він часто бере участь у процесі злиття та придбання. В основному HR розглядають як функцію підтримки бізнесу, допомагаючи мінімізувати витрати та зменшити ризик [17].

РОЗДІЛ 2

ОСОБЛИВОСТІ ВПРОВАДЖЕННЯ СУЧАСНИХ УПРАВЛІНСЬКИХ ТЕГНОЛОГІЇ

2.1 Огляд літератури

Сучасний етап розвитку українського суспільства характеризується необхідністю забезпечення надійності функціонування всіх секторів економіки, що залежить у першу чергу від готовності й спроможності персоналу на належному рівні виконувати свої професійні обов'язки. У зв'язку з цим особливого значення набуває завдання створення ефективної системи забезпечення надійності персоналу.

Відсутність теоретично обґрунтованої та практично апробованої системи, яка б враховувала особливості соціально-економічної та політичної ситуації в суспільстві, негативно відбивається на виконанні головного завдання системи управління персоналом – забезпечення високого рівня професіоналізму та надійності персоналу підприємств.

Вирішити це завдання можливо лише шляхом використання ефективної технології відбору, збереження та розвитку персоналу з метою підвищення якісних характеристик та забезпечення надійності кожного з працівників. Тому на сьогодні в умовах вкрай актуальними є вдосконалення існуючих технологій впливу на персонал підприємства.

Технологія управління персоналом підприємства покликана оптимізувати процес управління, завдяки раціональних методів, операцій та процедур прийняття управлінських рішень з метою ефективного впливу на людську складову підприємства.

Дослідження виробничого та управлінського процесів, які відбуваються на підприємствах доводять, що процес управління технологічно можна поділити на три основні цикли:

1. Інформаційний цикл, який передбачає пошук, збирання, оброблення, передачу і зберігання науково-технічної, економічної, оперативновиробничої,

облікової та іншої інформації. Ці роботи здебільшого виконують фахівці та технічні виконавці.

2. Логіко-розумовий цикл, сутність якого полягає у розробленні та прийнятті управлінських рішень (дослідження, удосконалення технікоекономічних рішень та прогнозів). Ці роботи виконують фахівці та функціональні менеджери підприємств.

3. Організаційний цикл, який передбачає організаційний вплив на об’єкт управління з метою реалізації управлінського рішення (пошук і залучення персоналу, визначення посадових обов’язків, доведення завдань до виконавців, організація трудового процесу персоналу, оперативне управління, координація, стимулювання, зворотний зв’язок). Ці роботи виконують лінійні менеджери [18].

Результати досліджень свідчать, що у більшості випадків на підприємствах із розбалансованими показниками діяльності та нестійким рівнем розвитку процес прийняття рішень в системі управління персоналом відбувається відповідно до набутого досвіду керівництва і носить інтуїтивний характер. Оскільки технологія управління персоналом є суттєвим інструментом успішного функціонування складових елементів системи управління персоналом, перелік означених недоліків свідчить про те, що технологія потребує вдосконалення, шляхом вивчення і розвитку її складових.

2.2 Технологія управління персоналом HR

На думку К. Поппера технології містять програму і методологію діяльності з вирішення конкретного завдання [19]. Технологізація управління передбачає, перш за все, дроблення управлінського процесу на окремі процедури та операції, з подальшою регламентацією виконання процедур і операцій [20].

Технологія управління персоналом – це послідовність взаємопов’язаних етапів, операцій, дій, що покликані спрямувати діяльність персоналу на реалізацію поставлених цілей. HR-менеджер є фахівцем з управління людськими ресурсами. У компанії HR-менеджер виконує функції, пов'язані з підбором персоналу, розробкою програм адаптації, мотивації, комплексної оцінки (атестації) і ін.

Коло обов'язків і функцій HR-менеджера сильно розширився. Сталося це тому, що керівники солідних компаній розуміють важливість кваліфікованого персоналу. Співробітники фірми стають людським ресурсом, більш значущим, ніж фінансові, інформаційні чи технічні ресурси компанії.

HR - це скорочення від "human resources", в перекладі з англійської це означає "людські ресурси". HR-менеджер працює безпосередньо з персоналом, здійснює підбір, адаптацію, оцінку, просування, навчання співробітників. А так само веде кадрове діловодство.

Одним із найважливіших процесів управління персоналом є забезпечення роботи HR з оптимальною швидкістю, кількістю та витратами. HR процеси повинні додати цінність для бізнес операцій. Функції, які відповідають за практику управління людьми, сприяє формуванню корпоративної культури та дає змогу змінювати проекти управління в організації. HR не може належним чином виконувати свою роль у бізнесі без правильного налаштування HR-процесів.

Останнім часом людські ресурси взагалі не були зосереджені на процесах. Процедури були складними, хаотичними та пропускали будь-яке суттєве вимірювання продуктивності та якості.

Людські ресурси - важливий бізнес партнер в організації. Однак це також функція управління. Для HR потрібно реалізувати правильний баланс між контрольним середовищем та свободою керівників та службовців діяти. Занадто великий контроль робить процеси повільними, дорогими та блокує інновації. HR розробляє процеси, що включають управління ризиками та дотримання закону та інших внутрішніх політик та правил. Однак гладка і ефективна робота з персоналом завжди повинна бути кінцевою метою без жодних компромісів. Жодна високоефективна стратегія управління персоналом не може існувати без управління процесом.

Загалом, управління процесами управління персоналом та процеси тісно пов'язані з вимірюванням людських ресурсів та рейтингом управління персоналом. Вимірювання ефективності та якості - це єдиний спосіб, коли людські ресурси можуть зосередитись на ключових питаннях та проблемах. Більше того,

HR може надавати докази того, як це додає цінності бізнесу щодня. Проста інформаційна панель управління людськими ресурсами може розповісти більше, ніж регулярна презентація для керівної команди. Це також допомагає людським ресурсам керувати операційними витратами на персонал. Він визначає правильні місця в організації HR, які можуть бути перероблені або вдосконалені.

Правильна конструкція HR-процесів сильно залежить від ролей та обов'язків персоналу та моделі HR. Лідер бізнесу встановлює бачення та напрямок організації. Команда лідерів визначає бізнес та кадрові стратегії. Роль керівника кадрів полягає у визначенні основних ролей та обов'язків з персоналу, включаючи операційну модель людських ресурсів. Це встановлює ще одну основу для управління персоналом процесів.

Сьогодні багато стартапів впроваджують прогресивні та сучасні практики управління людьми. Вони встановлюють нові стандарти; великі корпорації просто слідують за новими тенденціями та найкращими практиками. Ринок сьогодні є дуже конкурентоспроможним, і практика управління персоналом суттєво змінюється. Багато стартапів змінюють свій підхід до працівників, і вони встановлюють нові вищі стандарти та очікування. Кожен бізнес повинен стежити за розвитком найкращих практик управління персоналом та адаптувати внутрішню політику, щоб включити найкращі ідеї та розробити конкурентну перевагу. Жодна організація не може дозволити собі не оновлювати та змінювати свої HR-процеси.

2.3 Аналіз наявних рішень для HR

Кожен людський ресурс забезпечує декілька важливих бізнес процесів, які мають спільну мету - забезпечити бізнес і боротися з майбутніми викликами. Кожна компанія потребує ресурсів для доставки своєї продукції клієнтам. HR має переконатися, що організація має достатньо ресурсів, які ефективно розподіляються, і бізнес може доставляти інновації та вдосконалення своїм клієнтам.

Ключовим рушієм для кластеризації HR-процесів є модель HR. Це згода команди з управління персоналом, як вони розподіляють ролі та обов'язки

всередині організації з управління персоналом. Він визначає ключові внутрішні та зовнішні канали комунікації, як приймаються рішення та хто несе відповідальність за прийняття рішення.

1. Людські ресурси зазвичай кластеризують процеси в наступних функціях HR:
2. Дизайн організації та стратегічне планування робочої сили,
3. Підбір персоналу та персонал (включаючи набір соціальних медіа),
4. Бортове та нове орієнтування на прокат,
5. Управління продуктивністю,
6. Управління та розвиток талантів,
7. Навчання та розвиток,
8. Розвиток лідерства,
9. Управління життєвим циклом працівників, включаючи управління персоналом,
10. Розвиток персоналу та управління процесами управління персоналом,
11. Комунікація соціальних медіа [21].

Кожна успішна HR-організація починає всі процеси з дизайну організації. Це один із найстратегічніших процесів управління персоналом, оскільки він розробляє основні правила структуризації бізнесу. Він розробляє основні принципи організації, такі як проміжки та шари, необхідні навички та компетентності на кожному рівні та як процеси можуть безперебійно протікати через відділи та підрозділи. Це також є чудовою основою для стратегічного планування робочої сили. Більшість організацій не змінюють розподіл працівників на функції швидко. Однак ринок може істотно змінитися протягом ночі. Планування робочої сили керує оптимальним розподілом працівників, щоб бізнес працював плавно та ефективно. Коли все налагоджено, організація спритна і готова вирішити всі ринкові виклики.

Успішній організації потрібно найняти нові таланти з ринку праці та допомогти їм якомога швидше стати цінним членом команди. Підбір персоналу та персоналу орієнтовані на наймання на роботу всередині та зовні. Він починається

з результатів, розробок, принципів і правил, визначених маркетингом з персоналу, і закінчується успішним введенням у дію нового прокату. Йдеться про правильне управління витратами на підбір персоналу, суворе управління процесом та вимірювання ефективності.

Управління ефективністю зазвичай є найслабшим місцем серед кадрових процесів. Однак, це складний порядок денний, який гарантує, що всі працівники розуміють бачення, місію організації. Вони повинні отримати складний набір особистих цілей, які повністю відповідають бізнес-стратегії та цілям. Ефективне управління ефективністю стосується простого процесу встановлення цілей, регулярних відгуків про ефективність та офіційного оцінювання.

Компенсації та вигоди встановлюють стратегію та правила винагороди, щоб забезпечити досягнення внутрішнього справедливого капіталу та його правильний баланс. Він встановлює базову політику заробітної плати і гарантує, що всі керівники організації дотримуються правил. Він розробляє схеми стимулювання для центрів продажів та викликів. Зазвичай він контролює бюджет витрат на персонал і за потреби надсилає попередні попередження. Це створює політику виплат, щоб гарантувати, що працівники можуть використовувати блага, не передбачені прямими конкурентами.

Розвиток та управління талантами має чітке спрямування. У ньому викладені процеси та процедури, що визначають високих потенційних працівників, які повинні займати вищу посаду в організації в осяжному майбутньому. Розвиток лідерства має єдину мету - виробити нових лідерів для організації. Кожен керівник повинен визначити наступників, і вони повинні бути готові зайняти посаду, коли чинний керівник покине організацію. Він запускає конкретні процеси та процедури для виявлення прогалин та всіх найкращих талантів з різних частин організації. Він працює із суворо обмеженою групою працівників і готує їх до того, щоб взяти на себе керівну роль у майбутньому.

Життєвий цикл працівників - це процеси, орієнтовані на адміністрування персоналу, відстеження відвідуваності та обробку заробітної плати. Ці процеси є основою людських ресурсів, які повинні просто працювати. Команда повинна бути

зосереджена на досконалості та ефективності. Компанія може вижити без HR-стратегії, але вона не може існувати без надійної обробки оплати праці.

2.4 Категорії програмних рішень

Програмне забезпечення інформаційної системи управління персоналом (HRIS) - це загальний термін, що включає стільки різних типів програмного забезпечення, скільки існує компаній, що займаються розробкою програмного забезпечення для персоналу. Ці продукти відомі під різними назвами, але мета цих багатофункціональних систем полягає в тому, щоб зберігати індивідуальні дані про співробітників, управляти нарахуванням заробітної плати та адмініструванням.

Кількість часу, який менеджери по персоналу витрачають на роботу над ручним введенням даних, скоротилося, ці менеджери все ще витрачають до 50 відсотків свого часу на ручні завдання. За цим уважно стежать ручні коригування (30,6%) і витрачають час на обмін інформацією між нестандартизованими інструментами управління персоналом в організації (27,4%).

Системи HRIS призначені для оптимізації та автоматизації трьох традиційних стовпів управління персоналом: заробітної плати, часу і відвідуваності, а також управління перевагами. А сучасні рішення значно скорочують ручне введення даних. Компанії, які шукають правильне програмне забезпечення HRIS, можуть очікувати побачити ці функції:

1. Платіжна відомість,
2. Час відстеження,
3. Планування співробітників,
4. Управління перевагами,
5. Портал самообслуговування співробітників.

Інші включені функції в HRIS:

1. Наймання,
2. Відстеження заявника,
3. Облік,
4. Управління продуктивністю [22].

Підйом заявника систем спостереження та програмне забезпечення рекрутингу може бути безпосередньо пов'язаний з підвищенням автоматизацією багатьох традиційного введення даних і адміністративних завданнями, співробітниками відділу кадрів, використовуваними для заповнення днів. Тепер, коли відділ кадрів не витрачає більшу частину свого часу на ручний розрахунок тимчасових карт і відрахувань на заробітну плату, ці співробітники можуть зосередитися на залученні і наймання зіркових співробітників.

Системи відстеження кандидатів і програмне забезпечення для набору персоналу мають подібні функції, але часто різняться за масштабом. Програмне забезпечення рекрутингу найчастіше використовується агентствами з підбору персоналу та великими корпоративними компаніями з високими вимогами до роботи, оскільки вони створені для того, щоб справлятися з масовими або постійними зусиллями по підбору персоналу.

Системи відстеження кандидатів, з іншого боку, часто використовуються організаціями, яким потрібна відстеження життєвого циклу співробітників, і можуть включати модулі управління ефективністю роботи і залучення на додаток до традиційних функцій: дошка оголошень, резюме, співбесіди і адаптації.

РОЗДІЛ 3

HRIS - ІНФОРМАЦІЙНА СИСТЕМА ЛЮДСЬКИХ РЕСУРСІВ

3.1 Аналіз інформаційної системи

Інформаційна система людських ресурсів (HRIS) - це програмне забезпечення, яке забезпечує централізоване сховище головних даних працівників, необхідне групі управління людськими ресурсами (HR) для завершення основних процесів людських ресурсів.

HRIS зберігає, обробляє та керує даними про співробітників, такими як імена, адреси, національні посвідчення особи або номери соціального страхування, інформація про дозвіл на візу чи роботу та інформацію про утриманців. Зазвичай він також забезпечує функції HR, такі як підбір персоналу , відстеження заявників , управління часом та відвідуванням відвідувачів, оцінка ефективності та адміністрування пільг . Він може також містити функції самообслуговування працівників , а можливо, навіть функції бухгалтерського обліку.

У чомусь HRIS можна вважати розумною базою даних про працівників. Взаємодія даних, процеси, які можна виконати, та можливості звітності роблять дані, що зберігаються в системі, більш доступними та зручними.

3.2 Переваги HRIS

HRIS дозволяє відділу кадрів витрачати менше часу на діловодство, допомагає забезпечити точність даних працівників і може дати можливість працівникам брати більшу роль в управлінні їх інформацією.

Наявність централізованого сховища даних про співробітників усуває необхідність зберігання паперових файлів, які можуть бути легко пошкоджені, а також необхідність пошуку великих паперових файлів співробітників для пошуку інформації. Залежно від типу програмного забезпечення HRIS, воно повинно генерувати різні звіти, надавати можливості спеціальної звітності та пропонувати аналітику HR для важливих показників, таких як кількість рахунків та оборот. Сучасне програмне забезпечення HRIS також пропонує можливості

візуалізації даних про співробітників, такі як автоматично надані організаційні схеми [23].

Коли HRIS має службового або менеджерського самообслуговування, процес внесення основних даних працівника або організаційних змін стає більш ефективним і вимагає менше часу, ніж для запитів на паперовій основі. Робочі процеси затвердження дозволяють затверджувати або відхиляти зміни, автоматично повідомляючи про необхідних осіб. HRIS може також запропонувати мобільні можливості, що розширюють самообслуговування та надають додаткову гнучкість віддаленим працівникам.

3.3 Безпека та конфіденційність HRIS

HRIS також допомагає захищати дані співробітників і зберігати інформацію приватною. Під час використання паперових форм або електронних таблиць інформація легко може отримати доступ до людей, які можуть не мати повноважень доступу до неї. HRIS може захистити інформацію таким чином, що до неї можуть звертатися лише ті особи, які потребують доступу до неї.

Безпека даних та конфіденційність є важливими факторами при обробці конфіденційної особистої інформації, особливо в таких країнах, як Німеччина чи Франція, де робочі ради відіграють важливу роль у захисті даних працівників. За винятком блокування ключа, захист паперових записів може бути вкрай складним.

3.4 Типи програмного забезпечення HRIS

Різноманітні системи HRIS доступні та орієнтовані на різних типів клієнтів, починаючи від малого та середнього бізнесу аж до великих підприємств. Зазвичай різниця полягає в діапазоні та глибині особливостей для кожної області процесу.

Хоча більшість систем HRIS охоплюють значну частину описаних вище процесів, багато систем спрямованих на малі та середні підприємства, мають меншу глибину функціональності в кожній функції, ніж ті, що спрямовані на великі підприємства.

Таким чином ринок HRIS схожий на ринок автомобілів. Усі автомобілі отримають водія від А до В, але основні відмінності існують у якості та пропонованих зручностях.

3.5 HRIS функції

Як HR-інструмент, HRIS зазвичай містить модулі для вирішення наступних завдань:

1. Головне управління даними,
2. Організаційне управління, наприклад посади та відділи,
3. Самообслуговування працівників та менеджерів,
4. Відсутність та відпустка управління,
5. Переваги адміністрації,
6. Робочі процеси,
7. Оцінки ефективності,
8. Набір та відстеження заявників,
9. Управління компенсаціями,
10. Звітність та основна аналітика [24].

HRIS пропонує всебічний набір функцій управління людськими ресурсами для задоволення більшості потреб у персоналі. Без цього для зберігання даних потрібні незабезпечені або паперові документи або електронні таблиці. Введення даних вручну може спричинити помилки, а ручна перехресна перевірка документів і електронних таблиць може зайняти багато часу, а іноді і заплутатися, особливо при недостатній стандартизації способів збирання та зберігання даних.

Навіть коли придбана певна система для покриття процесу - наприклад, адміністрування пільг - це може означати ручне введення змін даних працівника, щоб оновити систему в курсі. Якщо використовується декілька систем, для кожної системи може знадобитися повторне введення даних, або користувачам може знадобитися експортувати дані з однієї системи, змінити їх і потім імпортувати в іншу систему.

У деяких випадках нарахування заробітної плати може бути частиною HRIS. Однак багато постачальників або не мають нарахування заробітної плати в рамках своїх пропозицій HRIS, або - як у Oracle , Workday та SAP SuccessFactors - вони продають фонд оплати праці як окрему систему, яка інтегрується з HRIS.

3.6 Аналітики HRIS

Аналітики HRIS - це висококваліфіковані професіонали, що володіють навичками IT та HR, які відповідають за управління HRIS та представляють відповідні та корисні дані про продуктивність праці, відвідуваність, навчання та оплату праці. Аналітики HRIS також забезпечують дотримання IT-підрозділів правил HR, а також надають необхідні ресурси працівникам та організовують відповідне оновлення обладнання. Великі організації можуть зайняти декількох аналітиків HRIS, щоб зосередитись на конкретних завданнях з персоналу, таких як виплати працівникам, компенсація або навчання.

Взагалі аналітики HRIS забезпечують ефективну організацію та подання інформації, що стосується всіх особливостей HR-функцій в компанії. Деякі конкретні переваги, які надають аналітикам HRIS, включають:

Обслуговування клієнтів як для співробітників користувачів HRIS, так і для користувачів управління [25].

Поради на основі аналізу процесів та результатів HRIS від того, хто спеціалізується на програмі та її виконанні.

Введення даних для великої кількості зібраної інформації про працівників.

Запевнення, що інформація та дані працівника зберігаються конфіденційними та безпечними.

Підвищена точність завдяки редагуванню та підтвердженню даних аналітиком до його повідомлення.

Наразі аналітикам HRIS не потрібно мати сертифікати. Однак, щоб бути конкурентоспроможними на ринку праці та збільшити зарплатний потенціал, кандидати пропонують надати доказ своєї досконалості у цій галузі та відданість персоналу, отримавши сертифікати.

РОЗДІЛ 4
РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ДЛЯ ОС IOS

4.1 Опис вимог до програмного забезпечення

Для демонстрації можливостей доповненої реальності, а також для того щоб дослідити та виявити її недоліки, було розроблено програмне забезпечення для операційної системи iOS. Для перевірки працездатність та самої розробки використовувся iPhone 8 Plus

Зважаючи область застосування та функціонування, мобільний додаток повинен викрнувати наступні функції:

розробка користувацького інтерфейсу

робота з даними мережі інтернет

обробка даних з мережі інтернет

коректна робота на пристроях з різною розрахунковою здатністю

інші умови (логування, та інше)

Було вирішено реалізовувати додаток з використанням архітектури MVC, використанням патернів проектування таких як Одинак, Делегування та Наглядач.

Данну архітектуру можна легко змінити та доповнити.

4.2 Розробка модулів додатку

Першим етапом розробики додатку стало створення його структури.Роботу додатку мржна продемонструвати на прикладі модулів, які реалізовують структуру роботи з об’єктами:

SceneDelegate–клас, який реалізує роботу навігації додатку

Coordinator – допоміжний клас, який регулює зміну вікон та правил навігації по додатку

Post – клас, в якому описана структура об’єкта “Пост”.

NewsViewController – клас в якому реалізована робота над об’єктами типу Post

Leaves- клас, в якому описана структура об’єкта “ Leaves”.

– LeavesViewController- клас, в якому реалізована робота над об’єктами типу Post та їх відображення в контролері.

Було вирішено реалізовувати додаток з використанням архітектури MVC, використанням патернів проектування таких як Одинак, Делегування та Наглядач.

Данну архітектуру можна легко змінити та доповнити.

Рисунок 4.1 – Схема навігації додатку

4.3 Розробка додатку

Додаток розроблений під ОС iOS з використанням мови програмування Swift. Список підтримуваних пристроїв наданий у додатку Б.

При запуску програмного забезпечення коростувача перекидає на екран логіну, продемонстровано на Рис. 4.2 та 4.3 . Для подальшого використання додатку йому потрібно увійти під своїм логіном та паролем або зареєструватись.

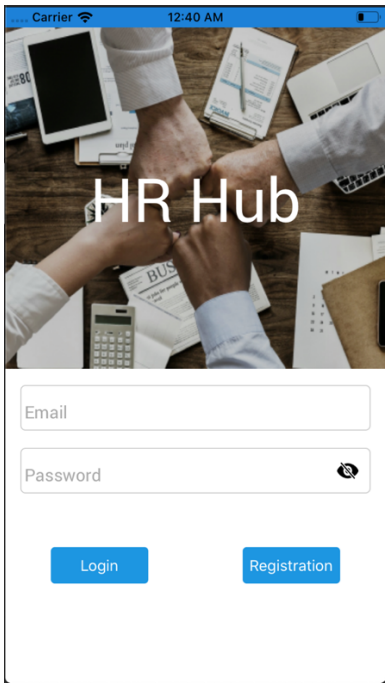

Рис. 4.2 – Вікно логіну

Рис 4.3– Вигляк екрану “Login” з заповненими полями

Після входу в додаток Користувачу показується головний екран на якому відображаються новини Рис4.4. З якими він може виконувати певні дії такі як вподобати запис та поділитись через стороні програми Рис 4.5

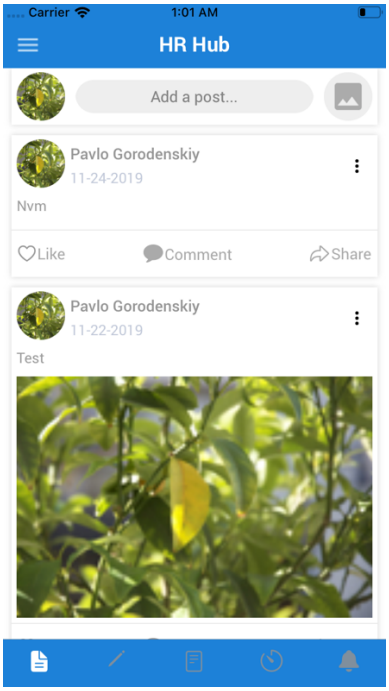

Рис 4.4 Вигляд головного вікна “News”

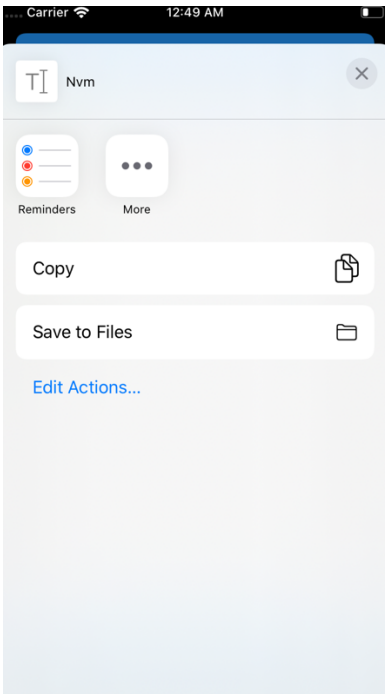

Рис.4.5 Вигляд вікна “Share”

З головного вікна Рис 4.4 відбувається навігація на потрібні користувачу вікна за допомогою нижньої та бокової меню Рис 4.6.

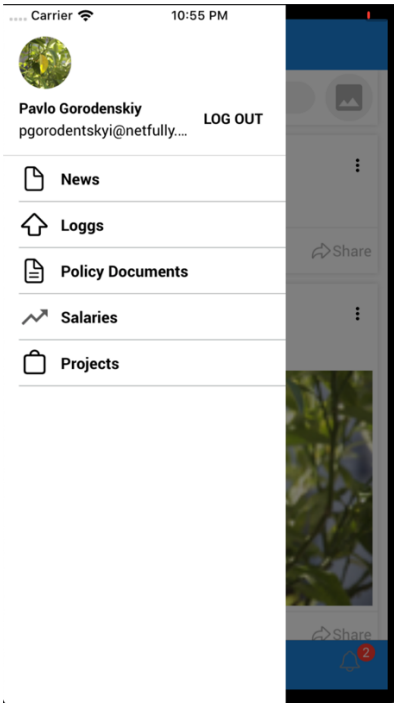

Рис 4.6 Вигляд вікна”Side menu”

Вікно “Leaves” відображає дані користувача відносно взятих ним вихідних та лікарняних. Він може переглянути залишок вихідних чи лікарняних днів, а також створити запит щоб отримати або використати лікарняні чи віхідні дні рис 4.7. Також користувач може переглянути статус поточних запитів.

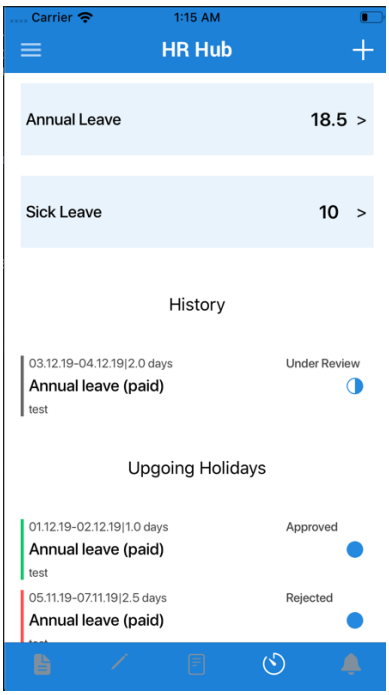

Рис 4.7 Вигляд вікнв “Leaves”

Користувач може створити свій власний пост використавши для цього вкладку “Post” Рис 4.8. На цьому вінкі користувач може додати текст до свого поста а також перейти до вікна вибору картинок та вибрати їх Рис 4.9

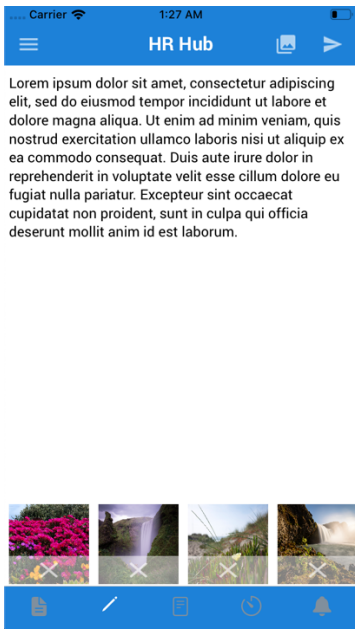

Рис 4.8 Вигляд вікна “Post”

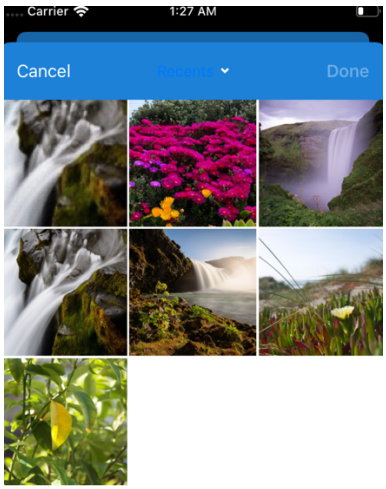

Рис 4.9 Вигляд вікна вибору картино “Image Picker”

В додатку реалізований функціонал, який дозволяє вести переписку між різними користувачами у яких є доступ до додатку Рис 4.10

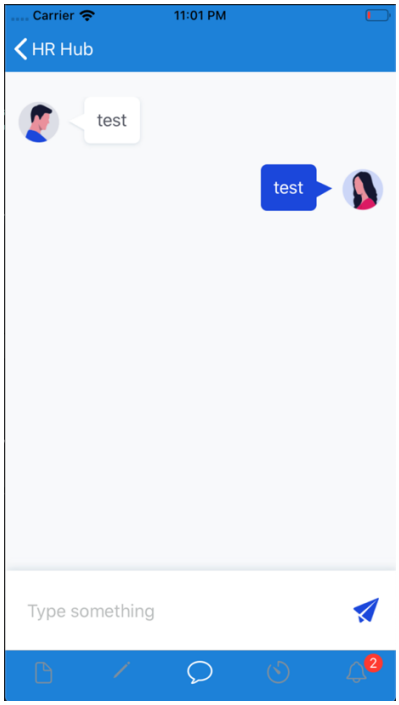

Рис 4.10 Вигляд вікна “Messages”

Оскільки додаток орієнтовани в більшості на айті компаніїї була додана можливість перегляду проектів на яких працює розробник та можливість перегляду актуальних на данний момент завдань Рис 4.11.

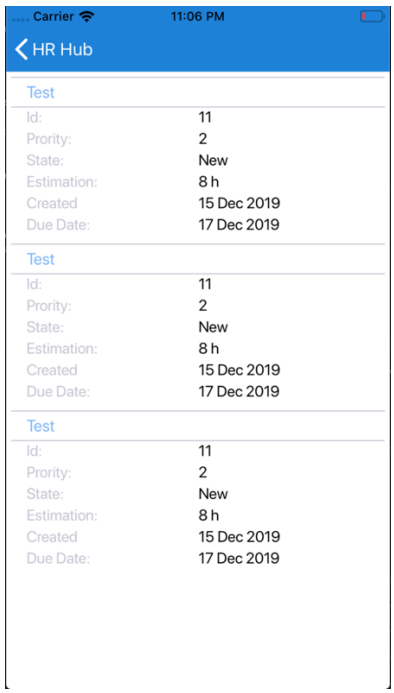

Рис 4.11 Винляд вікна “Tasks”

4.4 Опис використаних технологій

Для розробки програмного забезпечення було використано мову програмування Swift отсаньої пятої версії в середовищі для розробки Xcode від компанії Apple.

Для пришвидшення та полегшення контролю над життєвим циклом додатку ваикористовувалась архітектура програмування MVC.

Для роботи над користувацьким інтерфейсом додатку використовувався XML-файл з поєднанням бібліотеки UIKit.

В розробці використовувались стороні бібліотеки такі як:

Alamofire – бібліотека, яка полегшує роботу з мережею інтернет,

Gloss – бібліотека, яка полегшує роботу з JSON, XML та іншими варіантами відповіді сервера,

Photos – бібліотека, яка дозволяє працювати з камерою та збережини картинками на присторії,

SDWebImage – бібліотека, яка полегшує роботу з завантаженням та відображенням картинків на пристрої.

РОЗДІЛ 5
АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ

5.1 Застосування отриманих знань

Після проведених досліджень, було встановлено, що додаток є універсальним інструментом, яка поєднує управління робочою силою та можливостями управління людським капіталом (HCM) і підходить для підприємств усіх масштабів..

HR Hub - це єдине хмарне рішення, яке включає в себе підбір персоналу, на борту, управління ефективністю, планування компенсацій, час та відвідування, планування, управління відсутністю, нарахування заробітної плати тощо.

HR Hub містить один запис працівника, який ділиться у всіх його програмах. Усі дані співробітників зберігаються в єдиній базі даних, з єдиним користувальницьким інтерфейсом для витягування звітів та запуску робочих процесів по всій організації. Дані оновлюються в режимі реального часу, тому менеджери можуть приймати рішення на основі останньої, найточнішої інформації.

5.2 Проблеми, виявлені в ході досліджень

Однією основною проблемою, яку було зясовано в ході досліджень, являється дороговизна продутку. Що не зажди є доцільним для маленьких за розмірами компаній, оскільки віддавати значну долю прибутку за продукт.

Ще однією не мень важливою проблемою є не повністью готовий функціонал. Оскільки розробка HR-додатків являє собою тривалий та витратний процес, багато компаній випускають на ринок не повністью готовий продукт. Що є досить не зручним для користувачів. Досить часто буває таке, що додаток погано зберігає інформацію про процівника, та при підбиванні підсумків виводить не правдивий результат.

ВИСНОВКИ

В результаті досліджень, у дипломній роботі було проаналізовано структура та методи, що застосовуються в HR менеджменті.

Також було розроблено додаток, який наглядно демонструє усі фундаментальні приципи HR менеджменту.

Програма доволі зручна у використанні.

Тестування додатку показало його працездатність та відповідність технічним вимогам.

Програмне забезпечення містить наступний функціонал:

1. Створення нового посту,
2. Перегляд детальної інформації посту,
3. Можливість створення запиту вихідного чи лікарняного,
4. Пергляд використаних вихідних та лікарняних.

При виконанні роботи використовувався XCode – інтегроване середовище розробки (IDE) виробництва Apple. За допомогою якого можна створювати програмне забезпечення для різних платформ продукції Apple з використанням таких технологій

1. C ,
2. C++,
3. Swift,
4. Objective-C,
5. Ruby on Rails,
6. XML.

Paste json as code – середовище яке дозволяє легко опрацювати дані які приходять з сервера та працювати з ними.

Було проведено тестування додатку, яке продемонструвало підхід до тестування та його ефективність та зручність.

Недоліками IOS-додатку є:

1. Несумістність з старими версіями операційної системи IOS,

2. Не здастність працювати без підключення інтернету,

3. Важкість реалізаціх.

Дані дослідження можуть бути застосовані при розробці програмного забезпечення в інших сферах, розуміння її роботи, оцінці ризиків та можливостей такої розробки. Проведений аналіз виявив слабкі місця в розробці мобільних додатків, а також продемонстрував можливі рішення для вдосконалення програмного забезпечення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Lawler, E.E., Levenson, A. and Boudreau, J.W. (2004) HR Metrics and Analytics: Use and Impact. Human Resource Planning, 27, 27-35.
2. Джанет Марлер і Джон Будро характеризують HR аналітику. URL: https://www.trn.ua/articles/10174/
3. Jorrit van der Togt, Thomas Hedegaard Rasmussen (2017) Toward evidencebased HR // Journal of Organizational Effectiveness: People and Performance, Vol. 4 Issue: 2. – P. 127-132, DOI.org/10.1108/ JOEPP-02-2017-0013.
4. Манн Р. Контроллинг для начинающих. Система управления прибылью / Р. Манн Р., Е. Майер; пер с нем. – М.: Финансы и статистика, 2004.– 178 с.
5. Еронин В.А. Комплексная система диагностического анализа в управлении персоналом организации // Электронные публикации. № 18. – В.ГУ, 2007. URL: http://journal.vlsu.ru|index.php?id=735
6. Панченко, Г. С. Контролінг формування та використання людських ресурсів [Текст]: дис. ... кандидата екон. наук: 08.00.07 / Панченко Ганна Степанівна. – Донецьк, 2008. – 214 с.
7. Лукиных, Т. Н. Человеческие ресурсы в условиях информационной экономики [Електронний ресурс] / Т. Н. Лукиных // Открытый междисциплинарный электронный журнал «Гуманитарная информатика». – № 3. URL: http://huminf.tsu.ru/e-jurnal/magazine/3/ luk.htm.
8. Полозова, А. Н. Концептуальные положения персонал контроллинга в производственных организациях [Електронний ресурс] / А. Н. Полозова, С. В. Евсеева // Проблемы региональной экономики. URL: http://www.lerc.ru/?part=bulletin&art=19&page=13
9. Давидович І.Є. Контролінг : [навч. посібник] / І.Є. Давидович. – К. : Центр учбової літератури, 2008. – 552 с.

10. Михайлова А.В. Розвиток кадрового контролінгу в системі управління персоналом організації: дис. ... кандидата екон. наук : 08.00.05 / Михайлова Ганна Вікторівна. – М., 2006. – 210 с.
11. Бетанова И. Под прицелом кадровая стратегия. / И. Бетанова, О. Мищенко // Центр делового развития «Бизнес- град». URL: http:// www.biznes- grad.ru/library.html?articles&id=2
12. Уткін Е.А., Мірінюк І.В. Контролінг - М.: Фінанси і статистика, 1999. - 272 с.
13. Ковалев С.В. Система контроллинга персонала промышленной организации. – М.: КНОРУС, 2010. – 264 с.
14. Опитування MIT Sloan Management Review в партнерстві з IBM Institute for Business Value. URL: https://www935.ibm.com/services/uk/gbs/pdf/Analytics_The_new_path_to_value.pdf
15. Бондаренко Е. HR-Аналитика в Академии ДТЭК: проекты, которые меняют будущее / Е. Бондаренко // Академія. URL: ДТЕК, [Электронный ресурс], режим доступа: https://dtekacademy.com/blog/blog_id_88.
16. HR-аналітика в українських компаніях: реальність чи далеке майбутнє? URL: https://sumy.hh.ua/article/22927
17. Результати дослідження актуальності використання HR-аналітики в українських компаніях за результатами дослідження Академії ДТЕК. URL: https://app.powerbi.com/view?r=eyJrIjoiNGE4ZDc5YzYtMWQwNS00OWVhLTk2ZTctYWY1MjI0OTQ5MzhlIiwidCI6IjQ2ODI1NGFlLTlhYWQtNDZjOS1hMmEwLTcwYWU1NjU3MDUwZSIsImMiOjl9
18. People analytics – CIPD. URL: https://www.cipd.co.uk/Images/peopleanalytics-report_tcm18- 43755.pdf
19. Five most used HR analytics tools. URL: https://www.analyticsinhr.com/blog/hr-analytics-tools/
20. The 8 HR Analytics Every Manager Should Know About by Bernard Marr. URL: https://www.forbes.com/sites/bernardmarr/2016/03/01/the-8-hranalytics-every-manager-should-know-about/#57d7c1f6788f

21. Global Human Capital Trends. URL: https://www2.deloitte.com/insights/us/en/focus/human-capital-trends.html
22. Концепция контроллинга: Управленческий учет. Система отчетности. Бюджетирование / Horvart & Partners; Пер. с нем. – 3-е изд. – М.: Альпина Бизнес Букс, 2008. – 269 с.
23. As Dom Nicastro reports via CMSWire, “According to IBM’s 2017. URL: https://www.cmswire.com/customer-experience/what-is-customer-experiencemanagement/
24. HR Trends in 2019: The Future of Human Resource Management. URL: https://selecthub.com/hris/future-of-hr-software-trends/
25. Трубилин И. Т. Управление материальным стимулированием персонала в аграрных формированиях / И. Т. Трубилин, В.В. Говдя, Ж. В. Дегальцева // Политематический сетевой электронный научный журнал Кубанского государственного аграрного университета (Научный журнал КубГАУ). - [Электронный ресурс]. – Краснодар: КубГАУ, 2013. – №08(092). – IDA [article ID]: 0921308068. URL: http://ej.kubagro.ru/2013/08/pdf/68.pdf

ДОДАТКИ

ДОДАТОК А. Текст програми

```
import Foundation
import Gloss

class Post: NSObject, Glossy {
    let id, profileID: Int
    let body, dateCreated, creatorName: String
    let creatorProfileID: Int
    var expandet = false
    var vasLiked = false
    let creatorAltAttribute: String
    let pictureIDS: Int?
    var postPictures: [PostPicture]
    required init?(json: JSON) {
        self.id = "id" <~~ json ?? 0
        self.profileID = "profileID" <~~ json ?? 0
        self.creatorProfileID = "creatorProfileID" <~~ json ?? 0
        self.pictureIDS = "pictureIDS" <~~ json ?? 0
        self.body = "body" <~~ json ?? ""
        self.expandet = false
        self.dateCreated = "dateCreated" <~~ json ?? ""
        self.creatorName = "creatorName" <~~ json ?? ""
        self.creatorAltAttribute = "creatorAltAttribute" <~~ json ?? ""
        let postPicturesJSON: [JSON] = "postPictures" <~~ json ?? []
        self.postPictures = [PostPicture].from(jsonArray: postPicturesJSON) ?? []
    }

    func toJSON() -> JSON? {
        return jsonify([])
    }
}

struct PostPicture: Glossy {
    let postID, pictureID: Int
    let picture: Picture?
    init?(json: JSON) {
        self.postID = "postId" <~~ json ?? 0
        self.pictureID = "pictureId" <~~ json ?? 0
        self.picture = "picture" <~~ json
    }

    func toJSON() -> JSON? {
        return jsonify([])
    }
}

struct Picture: Glossy {
```

```
    let altAttribute: String
    init?(json: JSON) {
        self.altAttribute = "altAttribute" <~~ json ?? ""
    }

    func toJSON() -> JSON? {
        return jsonify([])
    }
} import Foundation

public struct Item {
    var name: String
    var detail: String

    public init(name: String, detail: String) {
        self.name = name
        self.detail = detail
    }
}
public class Leaves {
    var name: String
    var items: [Item]
    var collapsed: Bool

    public init(name: String, items: [Item], collapsed: Bool = true) {
        self.name = name
        self.items = items
        self.collapsed = collapsed
    }
}

public var sectionsData: [Leaves] = [
    Leaves(name: "Mac", items: [
        Item(name: "MacBook", detail: "Apple's ultraportable laptop, trading portability for speed and
connectivity.")
    ]),
    Leaves(name: "iPad", items: [
        Item(name: "iPad Pro", detail: "iPad Pro delivers epic power, in 12.9-inch and a new 10.5-inch
size.")
    ])
]
import Foundation
import Gloss

struct CoverImage: Glossy {
    let offset: Int
    let coverPicture: CoverPicture?
    init?(json: JSON) {
        offset = "offset" <~~ json ?? 0
        coverPicture = "picture" <~~ json
    }
```

```
    func toJSON() -> JSON? {
        return jsonify([])
    }
}

// MARK: - Picture

struct CoverPicture: Glossy {
    let altAttribute: String
    init?(json: JSON) {
        altAttribute = "altAttribute" <~~ json ?? ""
    }

    func toJSON() -> JSON? {
        return jsonify([])
    }
}

import Foundation
import Gloss


struct ProfileImage: Glossy {
    let imageUrl: String
    init?(json: JSON) {
        imageUrl = "imageUrl" <~~ json ?? ""
    }

    func toJSON() -> JSON? {
        return jsonify([])
    }
} import ImageSlideshow
import SDWebImage
import UIKit

class DetailNewsViewController: UIViewController {
    @IBOutlet var toolbarBackgroundViewBottomConstraint: NSLayoutConstraint!
    @IBOutlet var toolbarBackgroundView: UIView!

    @IBOutlet var toolBar: UIToolbar!
    @IBOutlet var commentTextField: UITextField!

    @IBOutlet var imageView: ImageSlideshow!
    @IBOutlet var imageViewHeightConstant: NSLayoutConstraint!
    @IBOutlet var bodyTextLabel: UILabel!
    @IBOutlet var creationDateLabel: UILabel!
    @IBOutlet var userNameLabel: UILabel!
    @IBOutlet var userImageView: UIImageView!
    @IBOutlet var moreActionButton: UIButton!

    @IBOutlet var likeButton: UIButton!
    @IBOutlet var shareButton: UIButton!
```

```
    var post: Post?
    var webImageSource: [SDWebImageSource] = []

    override func viewDidLoad() {
        super.viewDidLoad()
        hideKeyboardWhenTappedAround(viewController: self)
        commentTextField.inputAccessoryView = toolBar

//        NotificationCenter.default.addObserver(self, selector: #selector(handleKeyboardNotification),
name: UIResponder.keyboardWillShowNotification, object: nil)
//
//        NotificationCenter.default.addObserver(self, selector: #selector(handleKeyboardNotification),
name: UIResponder.keyboardWillHideNotification, object: nil)
        // Do any additional setup after loading the view.

        setupUi()
        setUpData()
    }

    func setupUi() {
        userImageView.layer.cornerRadius = userImageView.frame.size.width / 2
    }

    func setUpData() {
        bodyTextLabel.text = post!.body

        userNameLabel.text = post?.creatorName
        creationDateLabel.text = dateFormat(date: post!.dateCreated, FromDateFormat: "yyyy-MM-
dd'T'HH:mm:ss.SSSZ", toDateFormat: "MM-dd-yyyy")
        userImageView.sd_setImage(with: URL(string: post!.creatorAltAttribute), completed: nil)

        var webImageSource: [SDWebImageSource] = []

        if post!.vasLiked == true {
            likeButton.tintColor = UIColor.red
        } else {
            likeButton.tintColor = UIColor(red: 153 / 255, green: 153 / 255, blue: 153 / 255, alpha: 1)
        }
        if post!.postPictures.count != 0 {
            for item in post!.postPictures {
                let url = URL(string: item.picture!.altAttribute)
                webImageSource.append(SDWebImageSource(url: url!))
            }
        } else {
            imageViewHeightConstant.constant = 0
        }
        imageView.setImageInputs(webImageSource)
    }

    @IBAction func likeButtonAction(_ sender: Any) {
        registrationButton.layer.cornerRadius = 4
```

```
    }

    func login() {
        let progress = ProgressHUD(theme: .dark)
        view.addSubview(progress)
        view.isUserInteractionEnabled = false

        BackendModule.sharedInstance.postLogin(email: emailTextField.text ?? "", password: pass-
wordTextField.text ?? "", grant_type: "password", client_id: "teamnety_spa", success: {
            UserDefaults.standard.set(true, forKey: ConstantUserDefaultsKeys.status)

            self.view.isUserInteractionEnabled = true
            progress.hide()
            Coordinator().navigateToTabbarController(self)
        }) { error in

            self.view.isUserInteractionEnabled = true
            progress.hide()
            ErrorReporting.showMessage(title: "Error", msg: error, viewController: self)
            print(error)
        }
    }
}

extension LoginViewController: UITextFieldDelegate {
    func textFieldDidBeginEditing(_ textField: UITextField) {
        textField.layer.borderColor = UIColor(red: 28 / 255, green: 129 / 255, blue: 217 / 255, alpha:
1).cgColor
        textField.layer.borderWidth = 2
        textField.layer.cornerRadius = 5
    }

    func textFieldDidEndEditing(_ textField: UITextField) {
        textField.layer.borderColor = UIColor(red: 206 / 255, green: 206 / 255, blue: 206 / 255, alpha:
1).cgColor
        textField.layer.borderWidth = 1
        textField.layer.cornerRadius = 5
    }
}
import Photos
import SDWebImage
import UIKit

class CreatePostViewController: SideBaseViewController, UITextViewDelegate {
    @IBOutlet var textView: UITextView!
    @IBOutlet var textViewBottomConstraint: NSLayoutConstraint!

    var placeholderLabel: UILabel!
    var ids: [Int] = [Int]()
    var selectedAssets = [PHAsset]()
    var usedPictureIDs: [Int] = [Int]()
    var isEdit = false
```

```
    var post: Post?
    let progress = ProgressHUD(theme: .dark)

    lazy var flowLayout: UICollectionViewFlowLayout = {
        let f = UICollectionViewFlowLayout()
        f.itemSize = CGSize(width: 85, height: 85)
        f.minimumLineSpacing = 10
        f.sectionInset = UIEdgeInsets(top: 0, left: 5, bottom: 0, right: 5)
        f.scrollDirection = UICollectionView.ScrollDirection.horizontal
        return f
    }()

    lazy var collection: UICollectionView = {
        let cv = UICollectionView(frame: CGRect(x: 0, y: (self.tabBarController?.tabBar.frame.origin.y)!
- 90, width: self.view.frame.size.width, height: 90), collectionViewLayout: self.flowLayout)
        cv.translatesAutoresizingMaskIntoConstraints = false
        cv.setCollectionViewLayout(self.flowLayout, animated: true)
        cv.backgroundColor = .white
        cv.dataSource = self
        cv.delegate = self
        cv.register(UINib(nibName: ConstantStoryboard.createPostImageCollectionViewCell, bundle:
nil), forCellWithReuseIdentifier: ConstantID.createPostCollectionViewCell)
        return cv
    }()

    override func viewDidLoad() {
        super.viewDidLoad()

        textView.delegate = self
        hideKeyboardWhenTappedAround(viewController: self)

        guard let addPhotoFromLibrary = UIImage(named: "ic_photo_library"),
            let send = UIImage(named: "ic_send") else { return }

        addRightBarButtonsWithImage(buttonImageFirst: addPhotoFromLibrary, buttonImageSecond:
send, actionForFirstButton: #selector(sendPost), actionForSecondButton: #selector(addPhoto))
        textView.addDoneButtonOnKeyboard()
    }

    override func viewWillAppear(_ animated: Bool) {
        super.viewWillAppear(animated)
        if isEdit == false {
            textView.text = "Post something..."
            textView.textColor = UIColor.lightGray
        } else {
            view.addSubview(collection)
            textView.text = post?.body
            collection.reloadData()
            textView.textColor = UIColor.black
        }
    }
```

```
    func textViewDidBeginEditing(_ textView: UITextView) {
        if textView.textColor == UIColor.lightGray {
            textView.text = nil
            textView.textColor = UIColor.black
        }
    }

    func textViewDidEndEditing(_ textView: UITextView) {
        if textView.text.isEmpty {
            textView.text = "Post something..."
            textView.textColor = UIColor.lightGray
        }
    }
}

extension CreatePostViewController: UICollectionViewDelegate, UICollectionViewDataSource {
    func numberOfSections(in collectionView: UICollectionView) -> Int {
        if post?.postPictures.count != nil {
            return 2
        }
        return 1
    }

    func collectionView(_ collectionView: UICollectionView, numberOfItemsInSection section: Int) ->
Int {
        if section == 0 && post?.postPictures.count != nil {
            return (post?.postPictures.count)!
        }
        return selectedAssets.count
    }

    func collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath: IndexPath) ->
UICollectionViewCell {
        let cell = collectionView.dequeueReusableCell(withReuseIdentifier: ConstantID.createPostCol-
lectionViewCell, for: indexPath) as! CreatePostImageCollectionViewCell
        if indexPath.section == 0 && post?.postPictures.count != nil {
            let image = post?.postPictures[indexPath.row]
            DispatchQueue.main.async {
                cell.selectedImageView.sd_setImage(with: URL(string: image?.picture?.altAttribute ?? ""),
completed: nil)
            }
        } else {
            let item = selectedAssets[indexPath.row]
            DispatchQueue.global(qos: .userInteractive).async {
                DispatchQueue.main.async {
                    cell.selectedImageView.image = GetImageFromAsset().getAssetImage(asset: item)
                }
            }
        }
        cell.delegate = self
        return cell
    }
```

```
}
import UIKit
protocol CreatePostImageCollectionViewCellDelegate {
   func deleteImageTapped(_ sender: CreatePostImageCollectionViewCell)
}

class CreatePostImageCollectionViewCell: UICollectionViewCell {
   @IBOutlet var selectedImageView: UIImageView!
   @IBOutlet var deleteButton: UIButton!
   var delegate: CreatePostImageCollectionViewCellDelegate?

   override func awakeFromNib() {
      super.awakeFromNib()
      // Initialization code
   }

   @IBAction func deleteButtonAction(_ sender: Any) {
      delegate?.deleteImageTapped(self)
   }
}
import BSImagePicker
import Photos
import UIKit

extension CreatePostViewController {
   @objc func sendPost() {
      guard let profileId = UserDefaults.standard.string(forKey: ConstantUserDefaultsKeys.profileId)
else { return }
      view.addSubview(progress)

      if isEdit == true {
         for item in (post?.postPictures)! {
            usedPictureIDs.append(item.pictureID)
         }
         sendPostApiCall(prifileId: profileId)
      }
      for item in selectedAssets {
         BackendModule.sharedInstance.postAddFile(image: GetImageFromAsset().getAssetImage(as-
set: item), success: { [unowned self] ids in
            self.ids.append(ids)
            if self.ids.count == self.selectedAssets.count {
               self.sendPostApiCall(prifileId: profileId)
               self.progress.hide()
            }
         }) { error in
            self.progress.hide()
            ErrorReporting.showMessage(title: "Error", msg: error, viewController: self)
         }
      }
   }

   func ressetTextViewData() {
```

```
        ids.removeAll()
        selectedAssets.removeAll()
        textView.text = ""
    }

    func sendPostApiCall(prifileId: String) {
        if isEdit == false {
            BackendModule.sharedInstance.postCreatePost(body: textView.text, profileId: prifileId, pic-
turesId: ids, success: {
                self.ressetTextViewData()
                self.progress.hide()
                self.collection.removeFromSuperview()
            }) { error in
                self.progress.hide()
                ErrorReporting.showMessage(title: "Error", msg: error, viewController: self)
            }
        } else {
            BackendModule.sharedInstance.postEditPost(body: textView.text, profileId: prifileId, id:
post!.id, picturesId: ids, usedPictureIDs: usedPictureIDs, success: {
                self.ressetTextViewData()
                self.progress.hide()

                self.collection.removeFromSuperview()
            }) { error in
                self.progress.hide()
                ErrorReporting.showMessage(title: "Error", msg: error, viewController: self)
            }
        }
    }

    @objc func addPhoto() {
        let vc = BSImagePickerViewController()
        let assets = selectedAssets
        let collection = PHAssetCollection.transientAssetCollection(with: assets, title: nil)
        let fetchedAssets = PHAsset.fetchAssets(in: collection, options: nil)

        selectedAssets.removeAll()
        vc.takePhotos = true
        vc.defaultSelections = fetchedAssets

        bs_presentImagePickerController(vc, animated: true,
                                        select: { (asset: PHAsset) -> Void in
                                            print("Selected: \(asset)")
                                        }, deselect: { (asset: PHAsset) -> Void in
                                            print("Deselected: \(asset)")
                                        }, cancel: { (assets: [PHAsset]) -> Void in
                                            print("Cancel: \(assets)")
                                        }, finish: { (assets: [PHAsset]) -> Void in
                                            print("Finish: \(assets)")
                                            //   self.textView.text = self.textView.text + "\n\n"
                                            if assets.count != 0 {
                                                self.view.addSubview(self.collection)
```

```
                    self.textViewBottomConstraint.constant = self.collec-
tion.frame.size.height
                } else {
                    self.collection.removeFromSuperview()
                }
                for item in assets {
                    self.selectedAssets.append(item)
                    //  self.addAtributedTextWithImage(item: item, inTextView:
self.textView)
                }
                self.collection.reloadData()
            }, completion: nil)
        }

        func addAtributedTextWithImage(item: PHAsset, inTextView: UITextView) {
            let attributeSting = NSMutableAttributedString(attributedString: inTextView.attributedText)
            let image1Attachment = NSTextAttachment()
            image1Attachment.image = GetImageFromAsset().getAssetThumbnail(asset: item)
            let image1String = NSAttributedString(attachment: image1Attachment)
            attributeSting.append(image1String)
            inTextView.attributedText = attributeSting
        }
    }
    import Foundation

    extension CreatePostViewController: CreatePostImageCollectionViewCellDelegate {
        func deleteImageTapped(_ sender: CreatePostImageCollectionViewCell) {
            guard let tappedIndexPath = collection.indexPath(for: sender) else { return }
            let rowNumber: Int = tappedIndexPath.row
            if tappedIndexPath.section == 0 && post?.postPictures.count != nil {
                post?.postPictures.remove(at: rowNumber)
            } else {
                selectedAssets.remove(at: rowNumber)
            }
            collection.reloadData()
            if selectedAssets.count == 0 && (post?.postPictures.count == 0 || post?.postPictures.count == nil)
    {
                collection.removeFromSuperview()
                textViewBottomConstraint.constant = 0
            }
        }
    }
    import BSImagePicker
    import CocoaDebug
    import ImageSlideshow
    import Photos
    import SDWebImage
    import UIKit

    class NewsViewController: SideBaseViewController, UIGestureRecognizerDelegate {
        @IBOutlet var tableView: UITableView!
        @IBOutlet var userProfileImageView: UIImageView!
```

```
    @IBOutlet var userNameLabel: UILabel!
    @IBOutlet var userEmailLabel: UILabel!
    @IBOutlet var coverProfileImageView: UIImageView!

    let NibNewsCell = UINib(nibName: ConstantStoryboard.newsTableViewCell, bundle: nil)
    let previousValue = 10
    var posts: [Post] = []
    var refreshControl = UIRefreshControl()
    var coverImage: CoverImage?
    var profileImage: ProfileImage?

    var selectedAssets = [PHAsset]()

    override func viewDidLoad() {
        super.viewDidLoad()
        tableView.delegate = self
        tableView.dataSource = self
        tableView.register(NibNewsCell, forCellReuseIdentifier: ConstantID.newsTableViewCell)

        refreshControl.addTarget(self, action: #selector(refresh), for: UIControl.Event.valueChanged)
        tableView.addSubview(refreshControl)

        userEmailLabel.text = UserDefaults.standard.string(forKey: ConstantUserDefaultsKeys.email)
        userNameLabel.text = String(format: "%@ %@", UserDefaults.standard.string(forKey: Constan-
tUserDefaultsKeys.firstName) ?? "", UserDefaults.standard.string(forKey: ConstantUserDe-
faultsKeys.lastName) ?? "")
        coverProfileImageView?.sd_setImage(with: URL(string: coverImage?.coverPicture?.altAttribute
?? ""), completed: nil)
        userProfileImageView?.sd_setImage(with: URL(string: UserDefaults.standard.string(forKey:
ConstantUserDefaultsKeys.avatarUrl) ?? ""), completed: nil)
        setupUI()
        getCoverImage()
        let tapUserImageView = UITapGestureRecognizer(target: self, action: #selector(addPhoto))
        tapUserImageView.delegate = self
        userProfileImageView.isUserInteractionEnabled = true
        userProfileImageView.addGestureRecognizer(tapUserImageView)
    }

    override func viewWillAppear(_ animated: Bool) {
        super.viewWillAppear(animated)
        getPosts()
        getCoverImage()
    }

    @objc func addPhoto() {
        let vc = BSImagePickerViewController()
        let assets = selectedAssets
        let collection = PHAssetCollection.transientAssetCollection(with: assets, title: nil)
        let fetchedAssets = PHAsset.fetchAssets(in: collection, options: nil)

        vc.takePhotos = true
        vc.defaultSelections = fetchedAssets
```

```
        vc.maxNumberOfSelections = 1
        bs_presentImagePickerController(vc, animated: true,
                                        select: { (asset: PHAsset) -> Void in
                                            print("Selected: \(asset)")
                                        }, deselect: { (asset: PHAsset) -> Void in
                                            print("Deselected: \(asset)")
                                        }, cancel: { (assets: [PHAsset]) -> Void in
                                            print("Cancel: \(assets)")
                                        }, finish: { (assets: [PHAsset]) -> Void in
                                            print("Finish: \(assets)")

                                            self.selectedAssets = assets
                                            self.tappedUserImageView()

        }, completion: nil)
    }

    func tappedUserImageView() {
        BackendModule.sharedInstance.postAddProfilePicture(image: GetImageFromAsset().getAs-
setImage(asset: selectedAssets[0]), success: { profile in

            self.profileImage = profile
            if let image = self.profileImage?.imageUrl {
                self.userProfileImageView?.sd_setImage(with: URL(string: image), completed: nil)
                UserDefaults.standard.set(image, forKey: ConstantUserDefaultsKeys.avatarUrl)
            }
            self.tableView.reloadData()

        }) { _ in
        }
    }

    func getCoverImage() {
        BackendModule.sharedInstance.getPrifileCovel(ProfileId: UserDefaults.standard.string(forKey:
ConstantUserDefaultsKeys.profileId) ?? "", success: { cover in
            self.coverImage = cover
            self.coverProfileImageView?.sd_setImage(with: URL(string: self.coverImage?.coverPic-
ture?.altAttribute ?? ""), completed: nil)
        }) { _ in
        }
    }
}

extension NewsViewController: UITableViewDelegate {
    func tableView(_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat {
        let post = posts[indexPath.row]
        if post.expandet == false && post.postPictures.count != 0 {
            return 400
        }
        if post.expandet == false && post.postPictures.count == 0 {
            if post.body.count < 130 {
                return UITableView.automaticDimension
```

```
        }
        return 200
      }

      return UITableView.automaticDimension
    }

    func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
      let itemStoryboard = UIStoryboard(name: ConstantStoryboard.main, bundle: nil)
      let vc = itemStoryboard.instantiateViewController(withIdentifier: ConstantID.newsDetailed) as!
DetailNewsViewController
      vc.post = posts[indexPath.row]
      navigationController?.show(vc, sender: nil)
    }
}

extension NewsViewController: UITableViewDataSource {
    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
      return posts.count
    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableView-
Cell {
      let cell = NewsTableViewCell.cellConfiguration(for: indexPath, with: tableView)

      cell.delegate = self
      let post = posts[indexPath.row]
      cell.textBodyLabel.text = post.body

      cell.userNameLabel.text = post.creatorName
      cell.dateLabel.text = dateFormat(date: post.dateCreated, FromDateFormat: "yyyy-MM-
dd'T'HH:mm:ss.SSSZ", toDateFormat: "MM-dd-yyyy")
      cell.userImageView.sd_setImage(with: URL(string: post.creatorAltAttribute), completed: nil)

      var webImageSource: [SDWebImageSource] = []

      if post.vasLiked == true {
        cell.likeButton.tintColor = UIColor.red
      } else {
        cell.likeButton.tintColor = UIColor(red: 153 / 255, green: 153 / 255, blue: 153 / 255, alpha: 1)
      }

      if post.postPictures.count != 0 {
        for item in post.postPictures {
          let url = URL(string: item.picture!.altAttribute)
          webImageSource.append(SDWebImageSource(url: url!))
        }
      } else {
        cell.imageSliderHeightConstraint.constant = 0
      }
      cell.imageSlider.setImageInputs(webImageSource)
```

```
        if cell.textBodyLabel.calculateMaxLines() < 3 {
            cell.showMoreButton.isHidden = true
            cell.showMoreButtonConstraint.constant = 0
        }

        return cell
    }
}
import Foundation
import UIKit

extension NewsViewController {
    @objc func refresh(sender: AnyObject) {
        guard let profileId = UserDefaults.standard.string(forKey: ConstantUserDefaultsKeys.profileId)
else { return }
        BackendModule.sharedInstance.getPosts(PreviousValue: previousValue, ProfileId: profileId,
size: BackendModule.sharedInstance.perPage, page: 1, success: { [unowned self] posts in
            self.posts = posts
            self.tableView.reloadData()
            self.refreshControl.endRefreshing()
        }) { error in
            ErrorReporting.showMessage(title: "Error", msg: error, viewController: self)
        }
    }

    func setupUI() {
        userProfileImageView.layer.cornerRadius = userProfileImageView.frame.size.width/2
    }

    func getPosts() {
        guard let profileId = UserDefaults.standard.string(forKey: ConstantUserDefaultsKeys.profileId)
else { return }
        let progress = ProgressHUD(theme: .dark)
        view.addSubview(progress)
        BackendModule.sharedInstance.getPosts(PreviousValue: previousValue, ProfileId: profileId,
size: BackendModule.sharedInstance.perPage, page: 1, success: { [unowned self] posts in
            self.posts = posts
            self.tableView.reloadData()
            progress.hide()
        }) { error in
            progress.hide()
            ErrorReporting.showMessage(title: "Error", msg: error, viewController: self)
        }
    }
}
import Foundation
import UIKit

extension NewsViewController: NewsTableViewCellDelegate {
    func showMoreTapped(_ sender: NewsTableViewCell) {
        guard let tappedIndexPath = tableView.indexPath(for: sender) else { return }
        let content = posts[tappedIndexPath.row]
```

```
        content.expandet = !content.expandet
        tableView.reloadRows(at: [tappedIndexPath], with: .none)
    }

    func shareTapped(_ sender: NewsTableViewCell) {
        guard let tappedIndexPath = tableView.indexPath(for: sender) else { return }
        let content = posts[tappedIndexPath.row]
        let textToShare = [content.body]
        let activityViewController = UIActivityViewController(activityItems: textToShare, application-
Activities: nil)
        activityViewController.popoverPresentationController?.sourceView = view
        activityViewController.excludedActivityTypes = [UIActivity.ActivityType.airDrop, UIActiv-
ity.ActivityType.postToFacebook]
        present(activityViewController, animated: true, completion: nil)
    }

    func commentTapped(_ sender: NewsTableViewCell) {
    }

    func likeTapped(_ sender: NewsTableViewCell) {
        guard let tappedIndexPath = tableView.indexPath(for: sender) else { return }
        let content = posts[tappedIndexPath.row]

        content.vasLiked = !content.vasLiked
        tableView.reloadData()
    }

    func menuTapped(_ sender: NewsTableViewCell) {
        let alert = UIAlertController(title: nil, message: nil, preferredStyle: .actionSheet)

        let editAction = UIAlertAction(title: "Edit", style: .default) {
            _ in

            let navController = self.tabBarController?.viewControllers![1] as! UINavigationController
            let vc = navController.topViewController as! CreatePostViewController
            guard let tappedIndexPath = self.tableView.indexPath(for: sender) else { return }
            let content = self.posts[tappedIndexPath.row]
            vc.post = content
            vc.isEdit = true
            self.tabBarController?.selectedIndex = 1
        }

        let deleteAction = UIAlertAction(title: "Delete", style: .default) {
            _ in
            guard let tappedIndexPath = self.tableView.indexPath(for: sender) else { return }
            let content = self.posts[tappedIndexPath.row]
            BackendModule.sharedInstance.deletePost(postId: content.id, success: {
                self.posts.remove(at: tappedIndexPath.row)
                self.tableView.beginUpdates()
                self.tableView.deleteRows(at: [tappedIndexPath], with: .fade)
                self.tableView.endUpdates()
            }) { _ in
```

```
            }
        }
        let cancelAction = UIAlertAction(title: "Cancel", style: .cancel) {
            _ in
        }

        alert.addAction(editAction)
        alert.addAction(deleteAction)
        alert.addAction(cancelAction)
        alert.popoverPresentationController?.sourceView = view
        present(alert, animated: true, completion: nil)
    }
}
import ImageSlideshow
import UIKit

protocol NewsTableViewCellDelegate {
    func showMoreTapped(_ sender: NewsTableViewCell)
    func shareTapped(_ sender: NewsTableViewCell)
    func commentTapped(_ sender: NewsTableViewCell)
    func likeTapped(_ sender: NewsTableViewCell)
    func menuTapped(_ sender: NewsTableViewCell)
}

class NewsTableViewCell: UITableViewCell {
    @IBOutlet var backgroundCellView: UIView!
    @IBOutlet var showMoreButton: UIButton!
    @IBOutlet var likeButton: UIButton!
    @IBOutlet var commentButton: UIButton!
    @IBOutlet var shareButton: UIButton!
    @IBOutlet var textBodyLabel: UILabel!
    @IBOutlet var dateLabel: UILabel!
    @IBOutlet var userNameLabel: UILabel!
    @IBOutlet var userImageView: UIImageView!
    @IBOutlet var menuButton: UIButton!
    @IBOutlet var imageSlider: ImageSlideshow!
    @IBOutlet var imageSliderHeightConstraint: NSLayoutConstraint!
    @IBOutlet var showMoreButtonConstraint: NSLayoutConstraint!

    var delegate: NewsTableViewCellDelegate?
    var images = [BundleImageSource]()

    override func awakeFromNib() {
        super.awakeFromNib()
        imageSlider.pageIndicatorPosition = .init(horizontal: .center, vertical: .bottom)
        imageSlider.contentScaleMode = UIViewContentMode.scaleAspectFill

        let pageControl = UIPageControl()
        pageControl.currentPageIndicatorTintColor = UIColor.lightGray
        pageControl.pageIndicatorTintColor = UIColor.black
        imageSlider.pageIndicator = pageControl
```

```
        imageSlider.activityIndicator = DefaultActivityIndicator()
        imageSlider.delegate = self

        userImageView.layer.cornerRadius = 23.5

        backgroundCellView.layer.shadowColor = UIColor.lightGray.cgColor
        backgroundCellView.layer.shadowOpacity = 0.5
        backgroundCellView.layer.shadowOffset = CGSize(width: 0, height: 0)
        backgroundCellView.layer.shadowRadius = 3
    }

    @IBAction func showMoreButtonAction(_ sender: Any) {
        delegate?.showMoreTapped(self)
    }

    @IBAction func likeButtonAction(_ sender: Any) {
        delegate?.likeTapped(self)
    }

    @IBAction func commentButtonAction(_ sender: Any) {
    }

    @IBAction func shareButtonAction(_ sender: Any) {
        delegate?.shareTapped(self)
    }

    @IBAction func menuButtonAction(_ sender: Any) {
        delegate?.menuTapped(self)
    }

    override func setSelected(_ selected: Bool, animated: Bool) {
        super.setSelected(selected, animated: animated)

        // Configure the view for the selected state
    }

    class func cellConfiguration(for indexPath: IndexPath, with tableView: UITableView) -> NewsTa-
bleViewCell {
        let cell = tableView.dequeueReusableCell(withIdentifier: ConstantID.newsTableViewCell, for:
indexPath) as! NewsTableViewCell
        return cell
    }
}

extension NewsTableViewCell: ImageSlideshowDelegate {
    func imageSlideshow(_ imageSlideshow: ImageSlideshow, didChangeCurrentPageTo page: Int) {
        print("current page:", page)
    }
}
import SDWebImage
import UIKit
```

```
class SidePanelViewController: UIViewController {
    @IBOutlet var headerImageView: UIImageView!
    @IBOutlet var userImageView: UIImageView!
    @IBOutlet var userNameLabel: UILabel!
    @IBOutlet var userEmailLAbel: UILabel!

    @IBOutlet var logoutButton: UIButton!
    @IBOutlet var tableView: UITableView!

    // TODO: create method for loading file from nib
    let NibSideMenu = UINib(nibName: ConstantStoryboard.leftMenuNibName, bundle: nil)
    let NibSideMenuWithoutIcon = UINib(nibName: ConstantStoryboard.leftMenuWithoutIcon-
NibName, bundle: nil)

    var nameArray = [["News", "Notification", "Policy Documents", "Core Behavior", "Calendar",
"Settings"], ["Share", "Send", "Action1", "Action2"]]

    override func viewDidLoad() {
        super.viewDidLoad()

        userImageView.layer.cornerRadius = 35

        userEmailLAbel.text = UserDefaults.standard.string(forKey: ConstantUserDefaultsKeys.email)
        userNameLabel.text = String(format: "%@ %@", UserDefaults.standard.string(forKey: Constan-
tUserDefaultsKeys.firstName) ?? "", UserDefaults.standard.string(forKey: ConstantUserDe-
faultsKeys.lastName) ?? "")

        tableView.delegate = self
        tableView.dataSource = self
        tableView.register(NibSideMenu, forCellReuseIdentifier: ConstantID.sideMenuCell)
        tableView.register(NibSideMenuWithoutIcon, forCellReuseIdentifier: ConstantID.side-
MenuWithoutIconCell)
    }

    override func viewWillAppear(_ animated: Bool) {
        super.viewWillAppear(animated)
        userImageView?.sd_setImage(with: URL(string: UserDefaults.standard.string(forKey: Constan-
tUserDefaultsKeys.avatarUrl) ?? ""), completed: nil)
    }

    @IBAction func logoutButtonAction(_ sender: Any) {
        removeAllFromUserDefaults()
        Coordinator().navigateToLoginViewController(self)
    }

    func changeMainViewController(to viewController: UIViewController) {
        let navigationViewController = UINavigationController(rootViewController: viewController)
        guard let slideMenuController = slideMenuController() else { return }
        slideMenuController.changeMainViewController(navigationViewController, close: true)
    }
}
```

```
extension SidePanelViewController: UITableViewDataSource {
   func numberOfSections(in tableView: UITableView) -> Int {
      return 2
   }

   func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
      if section == 0 {
         return 6
      } else {
         return 4
      }
   }

   func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableView-
Cell {
      if indexPath.section == 1 {
         let cell = tableView.dequeueReusableCell(withIdentifier: ConstantID.sideMenuWithoutIcon-
Cell, for: indexPath) as! SideMenuWithOutIconTableViewCell
         cell.actionNameLabel.text = nameArray[indexPath.section][indexPath.row]
         return cell
      } else {
         let cell = tableView.dequeueReusableCell(withIdentifier: ConstantID.sideMenuCell, for: in-
dexPath) as! SideMenuTableViewCell
         cell.actionLabel.text = nameArray[indexPath.section][indexPath.row]
         return cell
      }
   }
}

extension SidePanelViewController: UITableViewDelegate {
   func tableView(_ tableView: UITableView, heightForHeaderInSection section: Int) -> CGFloat {
      return 1
   }

   func tableView(_ tableView: UITableView, viewForHeaderInSection section: Int) -> UIView? {
      let headerView = UIView()
      headerView.backgroundColor = ConstantColor.sideMenuHeaderViewColor
      return headerView
   }

   func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
      if indexPath.section == 0 {
         switch indexPath.row {
         case 1:
            if let customTabBar = ((slideMenuController()?.mainViewController as? UINavigationCon-
troller)?.viewControllers.first as? TabbarViewController) {
               customTabBar.selectedIndex = 0
               changeMainViewController(to: customTabBar)
            }
         case 2:
            changeMainViewController(to: Coordinator().PolicyViewController())
         default:
```

```
            if let customTabBar = ((slideMenuController()?.mainViewController as? UINavigationCon-
troller)?.viewControllers.first as? TabbarViewController) {
                customTabBar.selectedIndex = 3
                changeMainViewController(to: customTabBar)
            }
        }
    } else {
        guard let customTabBar = storyboard?.instantiateViewController(withIdentifier: Constan-
tID.tabbar) as? TabbarViewController else {
            return
        }

        customTabBar.selectedIndex = 3
        changeMainViewController(to: customTabBar)
    }
  }
}
import UIKit

class SideBaseViewController: UIViewController {
  override func viewWillAppear(_ animated: Bool) {
    super.viewWillAppear(animated)
    setNavigationBarItem()
    changeStatusBarColor(color: ConstantColor.statusBarColor)
    addAtributedNAvigationBarTitle(title: "Teamnety")
  }
}
import UIKit

class SideMenuTableViewCell: UITableViewCell {
  @IBOutlet var actionImageView: UIImageView!
  @IBOutlet var actionLabel: UILabel!

  override func awakeFromNib() {
    super.awakeFromNib()
    // Initialization code
  }

  override func setSelected(_ selected: Bool, animated: Bool) {
    super.setSelected(selected, animated: animated)

    // Configure the view for the selected state
  }
}
import UIKit

class SideMenuWithOutIconTableViewCell: UITableViewCell {
  @IBOutlet var actionNameLabel: UILabel!
  override func awakeFromNib() {
    super.awakeFromNib()
    // Initialization code
  }
```

```
    override func setSelected(_ selected: Bool, animated: Bool) {
        super.setSelected(selected, animated: animated)

        // Configure the view for the selected state
    }
}
import Arale
import UIKit

class LeavesViewController: SideBaseViewController, UITableViewDataSource, UITableViewDele-
gate {
    let NibhHeaderViewCell = UINib(nibName: ConstantStoryboard.headerTableViewCell, bundle:
nil)
    let NibhLeavesCell = UINib(nibName: ConstantStoryboard.leavesTableViewCell, bundle: nil)

    @IBOutlet var tableView: UITableView!

    var leave = sectionsData

    let nib = UINib(nibName: "CollapsibleTableViewHeader", bundle: nil)

    override func viewDidLoad() {
        super.viewDidLoad()
        tableView.tableFooterView = UIView()

        tableView.register(nib, forHeaderFooterViewReuseIdentifier: "customSectionHeader")
        tableView.register(NibhHeaderViewCell, forCellReuseIdentifier: ConstantID.headerTableView-
Cell)
        tableView.register(NibhLeavesCell, forCellReuseIdentifier: ConstantID.leavesTableViewCell)

        tableView.delegate = self
        tableView.dataSource = self
    }

    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        if section == 0 || section == 1 {
            return leave[section].collapsed ? 0 : leave[section].items.count
        }
        return 5
    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableView-
Cell {
        if indexPath.section == 0 || indexPath.section == 1 {
            let cell = HeaderTableViewCell.cellConfiguration(for: indexPath, with: tableView)
            return cell
        } else {
            let cell = LeavesTableViewCell.cellConfiguration(for: indexPath, with: tableView)
            return cell
        }
    }
```

```
    func numberOfSections(in tableView: UITableView) -> Int {
        return 4
    }

    func tableView(_ tableView: UITableView, heightForHeaderInSection section: Int) -> CGFloat {
        return 90
    }

    func tableView(_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat {
        return UITableView.automaticDimension
    }

    func tableView(_ tableView: UITableView, viewForHeaderInSection section: Int) -> UIView? {
        if section == 0 || section == 1 {
            let view = self.tableView.dequeueReusableHeaderFooterView(withIdentifier: "customSection-
Header") as! CollapsibleTableViewHeader
            view.arrowLabel.text = ">"
            view.section = section
            view.setCollapsed(leave[section].collapsed)
            view.delegate = self
            if section == 0 {
                view.titleLabel.text = "Annual Leave"
            } else {
                view.titleLabel.text = "Sick Leave"
            }
            return view

        } else {
            let view = UIView()
            let textlabel = UILabel()
            let marginGuide = view.layoutMarginsGuide
            view.addSubview(textlabel)
            textlabel.textAlignment = .center
            textlabel.translatesAutoresizingMaskIntoConstraints = false
            textlabel.topAnchor.constraint(equalTo: marginGuide.topAnchor).isActive = true
            textlabel.trailingAnchor.constraint(equalTo: marginGuide.trailingAnchor).isActive = true
            textlabel.bottomAnchor.constraint(equalTo: marginGuide.bottomAnchor).isActive = true
            textlabel.leadingAnchor.constraint(equalTo: marginGuide.leadingAnchor).isActive = true
            if section == 3 {
                textlabel.text = "Upgoing Holidays"
            } else {
                textlabel.text = "History"
            }
            return view
        }
    }
}

extension LeavesViewController: CollapsibleTableViewHeaderDelegate {
    func toggleSection(_ header: CollapsibleTableViewHeader, section: Int) {
        let collapsed = !leave[section].collapsed
```

```
        leave[section].collapsed = collapsed
        header.setCollapsed(collapsed)

        tableView.reloadSections(NSIndexSet(index: section) as IndexSet, with: .automatic)
    }
}
import UIKit

class HeaderTableViewCell: UITableViewCell {

    override func awakeFromNib() {
        super.awakeFromNib()
        // Initialization code
    }

    override func setSelected(_ selected: Bool, animated: Bool) {
        super.setSelected(selected, animated: animated)

        // Configure the view for the selected state
    }

    class func cellConfiguration(for indexPath: IndexPath, with tableView: UITableView) ->
HeaderTableViewCell {
        let cell = tableView.dequeueReusableCell(withIdentifier: ConstantID.headerTableViewCell, for:
indexPath) as! HeaderTableViewCell
        return cell
    }

} import UIKit
protocol CollapsibleTableViewHeaderDelegate {
    func toggleSection(_ header: CollapsibleTableViewHeader, section: Int)
}

class CollapsibleTableViewHeader: UITableViewHeaderFooterView {
    var delegate: CollapsibleTableViewHeaderDelegate?
    var section: Int = 0
//    let backroundView = UIView()
//    let titleLabel = UILabel()
//    let countLabel = UILabel()
//    let arrowLabel = UILabel()
    @IBOutlet weak var titleLabel: UILabel!
    @IBOutlet weak var countLabel: UILabel!
    @IBOutlet weak var arrowLabel: UILabel!

//    override init(reuseIdentifier: String?) {
//        super.init(reuseIdentifier: reuseIdentifier)
//
////        let marginGuide = contentView.layoutMarginsGuide
////
////        contentView.addSubview(backgroundView ?? UIView())
////        backgroundView!.backgroundColor = UIColor(named: "LeavesHeadeViewSectionColor")
```

```
////        backgroundView!.translatesAutoresizingMaskIntoConstraints = false
////        backgroundView!.topAnchor.constraint(equalTo: marginGuide.topAnchor).isActive = true
////        backgroundView!.trailingAnchor.constraint(equalTo: marginGuide.trailingAnchor).isActive =
true
////        backgroundView!.bottomAnchor.constraint(equalTo: marginGuide.bottomAnchor).isActive =
true
////        backgroundView!.leadingAnchor.constraint(equalTo: marginGuide.leadingAnchor).isActive =
true
//
//       // Arrow label
////            contentView.addSubview(arrowLabel)
////            arrowLabel.textColor = UIColor.white
////            arrowLabel.translatesAutoresizingMaskIntoConstraints = false
////            arrowLabel.widthAnchor.constraint(equalToConstant: 12).isActive = true
////            arrowLabel.topAnchor.constraint(equalTo: marginGuide.topAnchor).isActive = true
////            arrowLabel.trailingAnchor.constraint(equalTo: marginGuide.trailingAnchor).isActive = true
////            arrowLabel.bottomAnchor.constraint(equalTo: marginGuide.bottomAnchor).isActive = true
////
////            // Title label
////            contentView.addSubview(titleLabel)
////            titleLabel.textColor = UIColor.white
////            titleLabel.translatesAutoresizingMaskIntoConstraints = false
////            titleLabel.topAnchor.constraint(equalTo: marginGuide.topAnchor).isActive = true
////            titleLabel.trailingAnchor.constraint(equalTo: marginGuide.trailingAnchor).isActive = true
////            titleLabel.bottomAnchor.constraint(equalTo: marginGuide.bottomAnchor).isActive = true
////            titleLabel.leadingAnchor.constraint(equalTo: marginGuide.leadingAnchor).isActive = true
//
//
//    }
//
//    required init?(coder aDecoder: NSCoder) {
//        fatalError("init(coder:) has not been implemented")
//    }

    override func awakeFromNib() {
        addGestureRecognizer(UITapGestureRecognizer(target: self, action: #selector(CollapsibleTa-
bleViewHeader.tapHeader(_:))))
    }

    @objc func tapHeader(_ gestureRecognizer: UITapGestureRecognizer) {
        guard let cell = gestureRecognizer.view as? CollapsibleTableViewHeader else {
            return
        }

        delegate?.toggleSection(self, section: section)
    }

    func setCollapsed(_ collapsed: Bool) {
        arrowLabel.rotate(collapsed ? 0.0 : .pi / 2)
    }
}
import UIKit
```

```
class LeavesTableViewCell: UITableViewCell {

    override func awakeFromNib() {
        super.awakeFromNib()
        // Initialization code
    }

    override func setSelected(_ selected: Bool, animated: Bool) {
        super.setSelected(selected, animated: animated)

        // Configure the view for the selected state
    }
    class func cellConfiguration(for indexPath: IndexPath, with tableView: UITableView) ->
LeavesTableViewCell {
        let cell = tableView.dequeueReusableCell(withIdentifier: ConstantID.leavesTableViewCell, for:
indexPath) as! LeavesTableViewCell
        return cell
    }
}
import UIKit

class ConstantID: NSObject {
    static let loginNavigation = "idLoginNavigationViewController"
    static let tabbar = "idTabbar"
    static let leftmenuViewController = "idLeftMenuViewController"
    static let sideMenuWithoutIconCell = "idSideMenuWithoutIconCell"
    static let newsTableViewCell = "idNewsTableViewCell"
    static let sideMenuCell = "idSideMenuCell"
    static let login = "idLogin"
    static let newsDetailed = "idNewsDetailed"
    static let createPostCollectionViewCell = "idImageCell"
    static let detailNewsTableViewCell = "idDetailNewsTableViewCell"
    static let PolicyViewController = "idPolicyViewController"

    static let calendarViewController = "idCalendarViewController"
    static let leavesViewController = "idLeavesViewController"
    static let headerTableViewCell =  "idHeaderTableViewCell"
    static let leavesTableViewCell = "idLeavesTableViewCell"
}
import Foundation

import UIKit

class ConstantStoryboard: NSObject {
    static let main = "Main"
    static let leftMenuNibName = "SideMenuTableViewCell"
    static let leftMenuWithoutIconNibName = "SideMenuWithOutIconTableViewCell"
    static let newsTableViewCell = "NewsTableViewCell"
    static let createPostImageCollectionViewCell = "CreatePostImageCollectionViewCell"
    static let detailNewsTableViewCell = "detailNewsTableViewCell"
    static let headerTableViewCell = "HeaderTableViewCell"
```

```
    static let leavesTableViewCell = "LeavesTableViewCell"
}
import Foundation
import UIKit
// TODO: use color Set
class ConstantColor:NSObject{
    static let navigationBarColor = UIColor(red: 28/255, green: 129/255, blue: 217/255, alpha: 1)
    static let statusBarColor = UIColor(red: 28/255, green: 129/255, blue: 217/255, alpha: 1)
    static let tabBarColor = UIColor(red: 28/255, green: 129/255, blue: 217/255, alpha: 1)
    static let toolBarColor = UIColor(red: 28/255, green: 129/255, blue: 217/255, alpha: 1)
    static let sideMenuHeaderViewColor = UIColor(red: 220/255, green: 220/255, blue: 220/255, al-
pha: 1)
}
import Foundation

class ConstantUserDefaultsKeys: NSObject {
    static let status = "status"
    static let access_token = "access_token"
    static let refresh_token = "refresh_token"
    static let email = "email"
    static let firstName = "firstName"
    static let avatarUrl = "avatarUrl"
    static let lastName = "lastName"
    static let profileId = "profileId"
}
import Foundation

class DecodeBase64URL: NSObject {
    func decode(jwtToken jwt: String) -> [String: Any] {
        let segments = jwt.components(separatedBy: ".")
    open override var supportedInterfaceOrientations : UIInterfaceOrientationMask {
        if let mainController = self.mainViewController{
            return mainController.supportedInterfaceOrientations
        }
        return UIInterfaceOrientationMask.all
    }

    open override var shouldAutorotate : Bool {
        return mainViewController?.shouldAutorotate ?? false
    }

    open override func viewWillLayoutSubviews() {
        // topLayoutGuideの値が確定するこのタイミングで各種ViewControllerをセットする
        setUpViewController(mainContainerView, targetViewController: mainViewController)
        setUpViewController(leftContainerView, targetViewController: leftViewController)
        setUpViewController(rightContainerView, targetViewController: rightViewController)
    }

    open override var preferredStatusBarStyle: UIStatusBarStyle {
        return self.mainViewController?.preferredStatusBarStyle ?? .default
    }
```

```
open override func openLeft() {
    guard let _ = leftViewController else { // If leftViewController is nil, then return
        return
    }

    self.delegate?.leftWillOpen?()

    setOpenWindowLevel()
    // for call viewWillAppear of leftViewController
    leftViewController?.beginAppearanceTransition(isLeftHidden(), animated: true)
    openLeftWithVelocity(0.0)

    track(.leftTapOpen)
}

open override func openRight() {
    guard let _ = rightViewController else { // If rightViewController is nil, then return
        return
    }

    self.delegate?.rightWillOpen?()

    setOpenWindowLevel()
    rightViewController?.beginAppearanceTransition(isRightHidden(), animated: true)
    openRightWithVelocity(0.0)

    track(.rightTapOpen)
}

open override func closeLeft() {
    guard let _ = leftViewController else { // If leftViewController is nil, then return
        return
    }

    self.delegate?.leftWillClose?()

    leftViewController?.beginAppearanceTransition(isLeftHidden(), animated: true)
    closeLeftWithVelocity(0.0)
    setCloseWindowLevel()
}

open override func closeRight() {
    guard let _ = rightViewController else { // If rightViewController is nil, then return
        return
    }

    self.delegate?.rightWillClose?()

    rightViewController?.beginAppearanceTransition(isRightHidden(), animated: true)
    closeRightWithVelocity(0.0)
    setCloseWindowLevel()
}
```

```
    open func addLeftGestures() {

        if leftViewController != nil {
            if SlideMenuOptions.panGesturesEnabled {
                if leftPanGesture == nil {
                    leftPanGesture = UIPanGestureRecognizer(target: self, action: #selector(self.handleLeft-
PanGesture(_:)))
                    leftPanGesture!.delegate = self
                    view.addGestureRecognizer(leftPanGesture!)
                }
            }

            if SlideMenuOptions.tapGesturesEnabled {
                if leftTapGesture == nil {
                    leftTapGesture = UITapGestureRecognizer(target: self, action: #selector(self.toggleLeft))
                    leftTapGesture!.delegate = self
                    view.addGestureRecognizer(leftTapGesture!)
                }
            }
        }
    }

    open func addRightGestures() {

        if rightViewController != nil {
            if SlideMenuOptions.panGesturesEnabled {
                if rightPanGesture == nil {
                    rightPanGesture = UIPanGestureRecognizer(target: self, action: #selector(self.han-
dleRightPanGesture(_:)))
                    rightPanGesture!.delegate = self
                    view.addGestureRecognizer(rightPanGesture!)
                }
            }

            if SlideMenuOptions.tapGesturesEnabled {
                if rightTapGesture == nil {
                    rightTapGesture = UITapGestureRecognizer(target: self, action: #selector(self.tog-
gleRight))
                    rightTapGesture!.delegate = self
                    view.addGestureRecognizer(rightTapGesture!)
                }
            }
        }
    }

    open func removeLeftGestures() {

        if leftPanGesture != nil {
            view.removeGestureRecognizer(leftPanGesture!)
            leftPanGesture = nil
```

```
    }

    if leftTapGesture != nil {
        view.removeGestureRecognizer(leftTapGesture!)
        leftTapGesture = nil
    }
}

open func removeRightGestures() {

    if rightPanGesture != nil {
        view.removeGestureRecognizer(rightPanGesture!)
        rightPanGesture = nil
    }

    if rightTapGesture != nil {
        view.removeGestureRecognizer(rightTapGesture!)
        rightTapGesture = nil
    }
}

open func isTagetViewController() -> Bool {
    // Function to determine the target ViewController
    // Please to override it if necessary
    return true
}

open func track(_ trackAction: TrackAction) {
    // function is for tracking
    // Please to override it if necessary
}

struct LeftPanState {
    static var frameAtStartOfPan: CGRect = CGRect.zero
    static var startPointOfPan: CGPoint = CGPoint.zero
    static var wasOpenAtStartOfPan: Bool = false
    static var wasHiddenAtStartOfPan: Bool = false
    static var lastState : UIGestureRecognizer.State = .ended
}

@objc func handleLeftPanGesture(_ panGesture: UIPanGestureRecognizer) {

    if !isTagetViewController() {
        return
    }

    if isRightOpen() {
        return
    }

    switch panGesture.state {
        case UIGestureRecognizerState.began:
```

```
            if LeftPanState.lastState != .ended &&  LeftPanState.lastState != .cancelled &&  Left-
PanState.lastState != .failed {
                return
            }

            if isLeftHidden() {
                self.delegate?.leftWillOpen?()
            } else {
                self.delegate?.leftWillClose?()
            }

            LeftPanState.frameAtStartOfPan = leftContainerView.frame
            LeftPanState.startPointOfPan = panGesture.location(in: view)
            LeftPanState.wasOpenAtStartOfPan = isLeftOpen()
            LeftPanState.wasHiddenAtStartOfPan = isLeftHidden()

            leftViewController?.beginAppearanceTransition(LeftPanState.wasHiddenAtStartOfPan, ani-
mated: true)
            addShadowToView(leftContainerView)
            setOpenWindowLevel()
        case UIGestureRecognizerState.changed:
            if LeftPanState.lastState != .began && LeftPanState.lastState != .changed {
                return
            }

            let translation: CGPoint = panGesture.translation(in: panGesture.view!)
            leftContainerView.frame = applyLeftTranslation(translation, toFrame:
            if let strongSelf = self {
                strongSelf.removeShadow(strongSelf.rightContainerView)
                strongSelf.enableContentInteraction()
                strongSelf.rightViewController?.endAppearanceTransition()
                strongSelf.delegate?.rightDidClose?()
            }
    }
  }
  open override func toggleLeft() {
    if isLeftOpen() {
        closeLeft()
        setCloseWindowLevel()
        // Tracking of close tap is put in here. Because closeMenu is due to be call even when the menu
tap.

        track(.leftTapClose)
    } else {
        openLeft()
    }
  }

  open func isLeftOpen() -> Bool {
    return leftViewController != nil && leftContainerView.frame.origin.x == 0.0
  }
```

```
    open func isLeftHidden() -> Bool {
        return leftContainerView.frame.origin.x <= leftMinOrigin()
    }

    open override func toggleRight() {
        if isRightOpen() {
            closeRight()
            setCloseWindowLevel()

            // Tracking of close tap is put in here. Because closeMenu is due to be call even when the menu
tap.
            track(.rightTapClose)
        } else {
            openRight()
        }
    }

    open func isRightOpen() -> Bool {
        return rightViewController != nil && rightContainerView.frame.origin.x == view.bounds.width
- rightContainerView.frame.size.width
    }

    open func isRightHidden() -> Bool {
        return rightContainerView.frame.origin.x >= view.bounds.width
    }

    open func changeMainViewController(_ mainViewController: UIViewController,  close: Bool) {

        removeViewController(self.mainViewController)
        self.mainViewController = mainViewController
        setUpViewController(mainContainerView, targetViewController: mainViewController)
        if close {
            closeLeft()
            closeRight()
        }
    }

    open func changeLeftViewWidth(_ width: CGFloat) {

        SlideMenuOptions.leftViewWidth = width
        var leftFrame: CGRect = view.bounds
        leftFrame.size.width = width
        leftFrame.origin.x = leftMinOrigin()
        let leftOffset: CGFloat = 0
        leftFrame.origin.y = leftFrame.origin.y + leftOffset
        leftFrame.size.height = leftFrame.size.height - leftOffset
        leftContainerView.frame = leftFrame
    }

    open func changeRightViewWidth(_ width: CGFloat) {

        SlideMenuOptions.rightBezelWidth = width
```

```
        var rightFrame: CGRect = view.bounds
        rightFrame.size.width = width
        rightFrame.origin.x = rightMinOrigin()
        let rightOffset: CGFloat = 0
        rightFrame.origin.y = rightFrame.origin.y + rightOffset
        rightFrame.size.height = rightFrame.size.height - rightOffset
        rightContainerView.frame = rightFrame
    }

    open func changeLeftViewController(_ leftViewController: UIViewController, closeLeft:Bool) {

        removeViewController(self.leftViewController)
        self.leftViewController = leftViewController
        setUpViewController(leftContainerView, targetViewController: leftViewController)
        if closeLeft {
            self.closeLeft()
        }
    }

    open func changeRightViewController(_ rightViewController: UIViewController, closeRight:Bool)
{
        removeViewController(self.rightViewController)
        self.rightViewController = rightViewController
        setUpViewController(rightContainerView, targetViewController: rightViewController)
        if closeRight {
            self.closeRight()
        }
    }

    fileprivate func leftMinOrigin() -> CGFloat {
        return  -SlideMenuOptions.leftViewWidth
    }

    fileprivate func rightMinOrigin() -> CGFloat {
        return view.bounds.width
    }


    fileprivate func panLeftResultInfoForVelocity(_ velocity: CGPoint) -> PanInfo {

        let thresholdVelocity: CGFloat = 1000.0
        let pointOfNoReturn: CGFloat = CGFloat(floor(leftMinOrigin())) + SlideMenuOptions.poin-
tOfNoReturnWidth
        let leftOrigin: CGFloat = leftContainerView.frame.origin.x

        var panInfo: PanInfo = PanInfo(action: .close, shouldBounce: false, velocity: 0.0)

        panInfo.action = leftOrigin <= pointOfNoReturn ? .close : .open

        if velocity.x >= thresholdVelocity {
            panInfo.action = .open
            panInfo.velocity = velocity.x
```

```
        } else if velocity.x <= (-1.0 * thresholdVelocity) {
            panInfo.action = .close
            panInfo.velocity = velocity.x
        }

        return panInfo
    }

    fileprivate func panRightResultInfoForVelocity(_ velocity: CGPoint) -> PanInfo {

        let thresholdVelocity: CGFloat = -1000.0
        let pointOfNoReturn: CGFloat = CGFloat(floor(view.bounds.width) - SlideMenuOptions.poin-
tOfNoReturnWidth)
        let rightOrigin: CGFloat = rightContainerView.frame.origin.x

        var panInfo: PanInfo = PanInfo(action: .close, shouldBounce: false, velocity: 0.0)

        panInfo.action = rightOrigin >= pointOfNoReturn ? .close : .open

        if velocity.x <= thresholdVelocity {
            panInfo.action = .open
            panInfo.velocity = velocity.x
        } else if velocity.x >= (-1.0 * thresholdVelocity) {
            panInfo.action = .close
            panInfo.velocity = velocity.x
        }

        return panInfo
    }

    fileprivate func applyLeftTranslation(_ translation: CGPoint, toFrame:CGRect) -> CGRect {

        var newOrigin: CGFloat = toFrame.origin.x
        newOrigin += translation.x

        let minOrigin: CGFloat = leftMinOrigin()
        let maxOrigin: CGFloat = 0.0
        var newFrame: CGRect = toFrame

        if newOrigin < minOrigin {
            newOrigin = minOrigin
        } else if newOrigin > maxOrigin {
            newOrigin = maxOrigin
        }

        newFrame.origin.x = newOrigin
        return newFrame
    }

    fileprivate func applyRightTranslation(_ translation: CGPoint, toFrame: CGRect) -> CGRect {

        var  newOrigin: CGFloat = toFrame.origin.x
```

```
        newOrigin += translation.x

        let minOrigin: CGFloat = rightMinOrigin()
        let maxOrigin: CGFloat = rightMinOrigin() - rightContainerView.frame.size.width
        var newFrame: CGRect = toFrame

        if newOrigin > minOrigin {
            newOrigin = minOrigin
        } else if newOrigin < maxOrigin {
            newOrigin = maxOrigin
        }

        newFrame.origin.x = newOrigin
        return newFrame
    }

    fileprivate func getOpenedLeftRatio() -> CGFloat {

        let width: CGFloat = leftContainerView.frame.size.width
        let currentPosition: CGFloat = leftContainerView.frame.origin.x - leftMinOrigin()
        return currentPosition / width
    }

    fileprivate func getOpenedRightRatio() -> CGFloat {

        let width: CGFloat = rightContainerView.frame.size.width
        let currentPosition: CGFloat = rightContainerView.frame.origin.x
        return -(currentPosition - view.bounds.width) / width
    }

    fileprivate func applyLeftOpacity() {

        let openedLeftRatio: CGFloat = getOpenedLeftRatio()
        let opacity: CGFloat = SlideMenuOptions.contentViewOpacity * openedLeftRatio
        opacityView.layer.opacity = Float(opacity)
    }


    fileprivate func applyRightOpacity() {
        let openedRightRatio: CGFloat = getOpenedRightRatio()
        let opacity: CGFloat = SlideMenuOptions.contentViewOpacity * openedRightRatio
        opacityView.layer.opacity = Float(opacity)
    }

    fileprivate func applyLeftContentViewScale() {
        let openedLeftRatio: CGFloat = getOpenedLeftRatio()
        let scale: CGFloat = 1.0 - ((1.0 - SlideMenuOptions.contentViewScale) * openedLeftRatio)
        let drag: CGFloat = SlideMenuOptions.leftViewWidth + leftContainerView.frame.origin.x

        SlideMenuOptions.contentViewDrag == true ? (mainContainerView.transform = CGAff-
ineTransform(translationX: drag, y: 0)) : (mainContainerView.transform = CGAffineTrans-
form(scaleX: scale, y: scale))
```

```
    }

    fileprivate func applyRightContentViewScale() {
        let openedRightRatio: CGFloat = getOpenedRightRatio()
        let scale: CGFloat = 1.0 - ((1.0 - SlideMenuOptions.contentViewScale) * openedRightRatio)
        let drag: CGFloat = rightContainerView.frame.origin.x - mainContainerView.frame.size.width

        SlideMenuOptions.contentViewDrag == true ? (mainContainerView.transform = CGAff-
ineTransform(translationX: drag, y: 0)) : (mainContainerView.transform = CGAffineTrans-
form(scaleX: scale, y: scale))
    }

    fileprivate func addShadowToView(_ targetContainerView: UIView) {
        targetContainerView.layer.masksToBounds = false
        targetContainerView.layer.shadowOffset = SlideMenuOptions.shadowOffset
        targetContainerView.layer.shadowOpacity = Float(SlideMenuOptions.shadowOpacity)
        targetContainerView.layer.shadowRadius = SlideMenuOptions.shadowRadius
        targetContainerView.layer.shadowPath = UIBezierPath(rect: targetCon-
tainerView.bounds).cgPath
    }

    fileprivate func removeShadow(_ targetContainerView: UIView) {
        targetContainerView.layer.masksToBounds = true
        mainContainerView.layer.opacity = 1.0
    }

    fileprivate func removeContentOpacity() {
        opacityView.layer.opacity = 0.0
    }


    fileprivate func addContentOpacity() {
        opacityView.layer.opacity = Float(SlideMenuOptions.contentViewOpacity)
    }

    fileprivate func disableContentInteraction() {
        mainContainerView.isUserInteractionEnabled = false
    }

    fileprivate func enableContentInteraction() {
        mainContainerView.isUserInteractionEnabled = true
    }

    fileprivate func setOpenWindowLevel() {
        if SlideMenuOptions.hideStatusBar {
            DispatchQueue.main.async(execute: {
                if let window = UIApplication.shared.keyWindow {
                    window.windowLevel = UIWindow.Level.statusBar + 1
                }
            })
        }
    }
```

```
    fileprivate func setCloseWindowLevel() {
        if SlideMenuOptions.hideStatusBar {
            DispatchQueue.main.async(execute: {
                if let window = UIApplication.shared.keyWindow {
                    window.windowLevel = UIWindow.Level.normal
                }
            })
        }
    }

    fileprivate func setUpViewController(_ targetView: UIView, targetViewController: UIViewCon-
troller?) {
        if let viewController = targetViewController {
            viewController.view.frame = targetView.bounds

            if (!children.contains(viewController)) {
                addChild(viewController)
                targetView.addSubview(viewController.view)
                viewController.didMove(toParent: self)
            }
        }
    }


    fileprivate func removeViewController(_ viewController: UIViewController?) {
        if let _viewController = viewController {
            _viewController.view.layer.removeAllAnimations()
            _viewController.willMove(toParent: nil)
            _viewController.view.removeFromSuperview()
            _viewController.removeFromParent()
        }
    }

    open func closeLeftNonAnimation(){
        setCloseWindowLevel()
        let finalXOrigin: CGFloat = leftMinOrigin()
        var frame: CGRect = leftContainerView.frame
        frame.origin.x = finalXOrigin
        leftContainerView.frame = frame
        opacityView.layer.opacity = 0.0
        mainContainerView.transform = CGAffineTransform(scaleX: 1.0, y: 1.0)
        removeShadow(leftContainerView)
        enableContentInteraction()
    }

    open func closeRightNonAnimation(){
        setCloseWindowLevel()
        let finalXOrigin: CGFloat = view.bounds.width
        var frame: CGRect = rightContainerView.frame
        frame.origin.x = finalXOrigin
        rightContainerView.frame = frame
```

```
        opacityView.layer.opacity = 0.0
        mainContainerView.transform = CGAffineTransform(scaleX: 1.0, y: 1.0)
        removeShadow(rightContainerView)
        enableContentInteraction()
    }

    // MARK: UIGestureRecognizerDelegate
    open func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer, shouldReceive touch:
UITouch) -> Bool {

        let point: CGPoint = touch.location(in: view)

        if gestureRecognizer == leftPanGesture {
            return slideLeftForGestureRecognizer(gestureRecognizer, point: point)
        } else if gestureRecognizer == rightPanGesture {
            return slideRightViewForGestureRecognizer(gestureRecognizer, withTouchPoint: point)
        } else if gestureRecognizer == leftTapGesture {
            return isLeftOpen() && !isPointContainedWithinLeftRect(point)
        } else if gestureRecognizer == rightTapGesture {
            return isRightOpen() && !isPointContainedWithinRightRect(point)
        }

        return true
    }

    // returning true here helps if the main view is fullwidth with a scrollview
    open func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer, shouldRecognizeSimul-
taneouslyWith otherGestureRecognizer: UIGestureRecognizer) -> Bool {
        return SlideMenuOptions.simultaneousGestureRecognizers
    }

    fileprivate func slideLeftForGestureRecognizer( _ gesture: UIGestureRecognizer, point:CGPoint) -
> Bool{
        return isLeftOpen() || SlideMenuOptions.panFromBezel && isLeftPointContainedWithinBez-
elRect(point)
    }

    fileprivate func isLeftPointContainedWithinBezelRect(_ point: CGPoint) -> Bool{
        if let bezelWidth = SlideMenuOptions.leftBezelWidth {
            var leftBezelRect: CGRect = CGRect.zero
            let tuple = view.bounds.divided(atDistance: bezelWidth, from: CGRectEdge.minXEdge)
            leftBezelRect = tuple.slice
            return leftBezelRect.contains(point)
        } else {
            return true
        }
    }

    fileprivate func isPointContainedWithinLeftRect(_ point: CGPoint) -> Bool {
        return leftContainerView.frame.contains(point)
    }
```

```
    fileprivate func slideRightViewForGestureRecognizer(_ gesture: UIGestureRecognizer,
withTouchPoint point: CGPoint) -> Bool {
        return isRightOpen() || SlideMenuOptions.rightPanFromBezel && isRightPointContainedWith-
inBezelRect(point)
    }

    fileprivate func isRightPointContainedWithinBezelRect(_ point: CGPoint) -> Bool {
        if let rightBezelWidth = SlideMenuOptions.rightBezelWidth {
            var rightBezelRect: CGRect = CGRect.zero
            let bezelWidth: CGFloat = view.bounds.width - rightBezelWidth
            let tuple = view.bounds.divided(atDistance: bezelWidth, from: CGRectEdge.minXEdge)
            rightBezelRect = tuple.remainder
            return rightBezelRect.contains(point)
        } else {
            return true
        }
    }

    fileprivate func isPointContainedWithinRightRect(_ point: CGPoint) -> Bool {
        return rightContainerView.frame.contains(point)
    }

}

extension UIViewController {

    public func slideMenuController() -> SlideMenuController? {
        var viewController: UIViewController? = self
        while viewController != nil {
            if viewController is SlideMenuController {
                return viewController as? SlideMenuController
            }
            viewController = viewController?.parent
        }
        return nil
    }

    public func addLeftBarButtonWithImage(_ buttonImage: UIImage) {
        let leftButton: UIBarButtonItem = UIBarButtonItem(image: buttonImage, style: UIBarButton-
Item.Style.plain, target: self, action: #selector(self.toggleLeft))
        navigationItem.leftBarButtonItem = leftButton
    }

    public func addRightBarButtonWithImage(_ buttonImage: UIImage) {
        let rightButton: UIBarButtonItem = UIBarButtonItem(image: buttonImage, style: UIBarButton-
Item.Style.plain, target: self, action: #selector(self.toggleRight))
        navigationItem.rightBarButtonItem = rightButton
    }

    @objc public func toggleLeft() {
```

```
        slideMenuController()?.toggleLeft()
    }

    @objc public func toggleRight() {
        slideMenuController()?.toggleRight()
    }

    @objc public func openLeft() {
        slideMenuController()?.openLeft()
    }

    @objc public func openRight() {
        slideMenuController()?.openRight()    }

    @objc public func closeLeft() {
        slideMenuController()?.closeLeft()
    }

    @objc public func closeRight() {
        slideMenuController()?.closeRight()
    }

    // Please specify if you want menu gesuture give priority to than targetScrollView
    public func addPriorityToMenuGesuture(_ targetScrollView: UIScrollView) {
        guard let slideController = slideMenuController(), let recognizers = slideController.view.ges-
tureRecognizers else {
            return
        }
        for recognizer in recognizers where recognizer is UIPanGestureRecognizer {
            targetScrollView.panGestureRecognizer.require(toFail: recognizer)
        }
    }
}

import Foundation
import UIKit

extension UITextView{
    @IBInspectable var doneAccessory: Bool{
        get{
            return self.doneAccessory
        }
        set (hasDone) {
            if hasDone{
                addDoneButtonOnKeyboard()
            }
        }
    }

    func addDoneButtonOnKeyboard()
    {
```

```
        let doneToolbar: UIToolbar = UIToolbar(frame: CGRect.init(x: 0, y: 0, width: UIS-
creen.main.bounds.width, height: 44))
        doneToolbar.barStyle = .default
        let flexSpace = UIBarButtonItem(barButtonSystemItem: .flexibleSpace, target: nil, action: nil)
        let done: UIBarButtonItem = UIBarButtonItem(title: "Done", style: .done, target: self, action:
#selector(self.doneButtonAction))

        let items = [flexSpace ,done]
        doneToolbar.items = items
        doneToolbar.sizeToFit()
        self.inputAccessoryView = doneToolbar
    }

    @objc func doneButtonAction()
    {
        self.resignFirstResponder()
    }
}
import Foundation
import UIKit
class ErrorReporting {
    class func showMessage(title: String, msg: String,viewController:UIViewController){
        let alert = UIAlertController(title: title, message: msg, preferredStyle: UIAlertControl-
ler.Style.alert)
        alert.addAction(UIAlertAction(title: "Ok", style: UIAlertAction.Style.default, handler: nil))
        viewController.present(alert, animated: true, completion: nil)
    }
}
import UIKit

protocol SBMessageInputViewDelegate: AnyObject {

    // TextViewDelegate methods
    func inputView(textView: UITextView, shouldChangeTextInRange: NSRange, replacementText:
String) -> Bool
    func inputViewDidChange(textView: UITextView)
    func inputViewDidBeginEditing(textView: UITextView)
    func inputViewShouldBeginEditing(textView: UITextView) -> Bool

    // Button tap callback methods
    func inputViewDidTapButton(button: UIButton)
}

@IBDesignable
class SBMessageInputView: UIView {

    @IBInspectable var placeholder: String = ""
    @IBInspectable var placeholderColor: UIColor = .lightGray
    @IBInspectable var textColor: UIColor = .black

    @IBInspectable var buttonImage: UIImage = SBMessageInputView.getDefaultImage() {
        didSet {
```

```
            guard let b = button else { return }
            b.setImage(buttonImage, for: .normal)
        }
    }

    @IBInspectable var viewBorderColor: UIColor = .gray {
        didSet {
            guard let mv = mainView else { return }
            mv.layer.borderColor = viewBorderColor.cgColor
        }
    }

    @IBInspectable var viewBorderWidth: CGFloat = 0.5 {
        didSet {
            guard let mv = mainView else { return }
            mv.layer.borderWidth = viewBorderWidth
        }
    }

    @IBInspectable var maxLines: CGFloat = 5.0

    @IBInspectable var textViewTopInset: CGFloat = 0.0 {
        didSet {
            guard let tv = textView else { return }
            tv.contentInset = UIEdgeInsets(top: textViewTopInset, left: tv.contentInset.left, bottom: tv.con-
tentInset.bottom, right: tv.contentInset.right)
        }
    }

    @IBInspectable var textViewLeftInset: CGFloat = 8.0 {
        didSet {
            guard let tv = textView else { return }
            tv.contentInset = UIEdgeInsets(top: tv.contentInset.top, left: textViewLeftInset, bottom: tv.con-
tentInset.bottom, right: tv.contentInset.right)
        }
    }

    var numberOfLines: CGFloat = 1.0 {
        didSet {
            if numberOfLines > oldValue && numberOfLines <= maxLines {
                increaseSize()
            } else if numberOfLines < oldValue && numberOfLines <= maxLines {
                reduceSize()
            } else {
                // do nothing
            }
        }
    }

    // Views
    var mainView: UIView?
    var textView: UITextView?
```

```
    var button: UIButton?
    var heightConstraint: NSLayoutConstraint?

    // Heights
    var originalViewHeight: CGFloat = 0.0
    var containerViewHeight: CGFloat {
        let top: CGFloat = 4.0
        let bottom: CGFloat = 4.0
        return originalViewHeight - (top + bottom)
    }
    var buttonViewHeight: CGFloat = 26.0
    var lineHeight: CGFloat = 0.0

    // Delegate
    var delegate: SBMessageInputViewDelegate?

    override init(frame: CGRect) {
        super.init(frame: frame)
        setupView()
    }

    required init?(coder aDecoder: NSCoder) {
        super.init(coder: aDecoder)
        setupView()
    }

    class func getDefaultImage()-> UIImage {
        if let defaultImage = UIImage(named: "send") {
            return defaultImage
        } else {
            return UIImage()
        }
    }

    override func layoutSubviews() {
        super.layoutSubviews()

        if originalViewHeight == 0 { originalViewHeight = frame.height }
        if let mv = mainView { mv.layer.cornerRadius = containerViewHeight / 2.0 }
        setTextView()
        if let tv = textView { tv.setContentOffset(CGPoint(x: -textViewLeftInset - 20, y: -textViewTop-
Inset), animated: false) }
    }

    fileprivate func setupView() {
        setContainerView()
    }

    func increaseSize() {

        let newConstant = containerViewHeight + ((numberOfLines - 1) * lineHeight) + 8.0
        if let c = heightConstraint {
```

```
            c.constant = newConstant
        } else {
            for constraint in constraints {
                if constraint.firstAttribute == .height {
                    constraint.constant = newConstant
                    break
                }
            }
        }
    }

    func reduceSize() {

        let newConstant = numberOfLines == 1 ? containerViewHeight + 8.0 : containerViewHeight +
((numberOfLines - 1) * lineHeight) + 8.0
        if let c = heightConstraint {
            c.constant = newConstant
        } else {
            for constraint in constraints {
                if constraint.firstAttribute == .height {
                    constraint.constant = newConstant
                    break
                }
            }
        }
    }

    fileprivate func setContainerView() {
        if mainView == nil {
            mainView = UIView(frame: .zero)
            guard let mainView = self.mainView else { return }
            addSubview(mainView)

            mainView.translatesAutoresizingMaskIntoConstraints = false
            let leadingConstraint = NSLayoutConstraint(item: mainView, attribute: .leading, relatedBy:
.equal, toItem: self, attribute: .leading, multiplier: 1, constant: 6)
            let trailingConstraint = NSLayoutConstraint(item: self, attribute: .trailing, relatedBy: .equal,
toItem: mainView, attribute: .trailing, multiplier: 1, constant: 6)
            let topConstraint = NSLayoutConstraint(item: mainView, attribute: .top, relatedBy: .equal, toI-
tem: self, attribute: .top, multiplier: 1, constant: 4)
            let bottomConstraint = NSLayoutConstraint(item: self, attribute: .bottom, relatedBy: .equal,
toItem: mainView, attribute: .bottom, multiplier: 1, constant: 4)

            addConstraints([topConstraint, bottomConstraint, leadingConstraint, trailingConstraint])
        }
    }

    fileprivate func setTextView() {

        if textView == nil {
            textView = UITextView(frame: .zero)
            button = UIButton(type: .custom)
```

```
        guard let button = self.button, let textView = self.textView, let mainView = self.mainView
else { return }
        textView.delegate = self

        if !placeholder.isEmpty {
           textView.text = placeholder
           textView.textColor = placeholderColor
        }

        button.setImage(buttonImage, for: .normal)
        button.addTarget(self, action: #selector(didTapButton(sender:)), for: .touchUpInside)

        textView.clipsToBounds = true
        textView.layer.cornerRadius = containerViewHeight / 2.0
        textView.contentInset = UIEdgeInsets(top: textViewTopInset, left: 0.0, bottom: 0.0, right: 0.0)
        textView.font = UIFont(name: "Roboto", size: 15)

        mainView.addSubview(textView)
        mainView.addSubview(button)

        textView.translatesAutoresizingMaskIntoConstraints = false
        textView.leadingAnchor.constraint(equalTo: mainView.leadingAnchor, constant:
textViewLeftInset).isActive = true
        textView.topAnchor.constraint(equalTo: mainView.topAnchor, constant: 0.0).isActive = true
        textView.bottomAnchor.constraint(equalTo: mainView.bottomAnchor, constant: 0.0).isActive
= true

        button.translatesAutoresizingMaskIntoConstraints = false
        button.heightAnchor.constraint(equalToConstant: buttonViewHeight).isActive = true
        button.widthAnchor.constraint(equalToConstant: buttonViewHeight).isActive = true
        button.leadingAnchor.constraint(equalTo: textView.trailingAnchor, constant: 5.0).isActive =
true
        button.trailingAnchor.constraint(equalTo: mainView.trailingAnchor, constant: -5.0).isActive =
true
        button.bottomAnchor.constraint(equalTo: mainView.bottomAnchor, constant: -5.5).isActive =
true
     }
   }

   @objc fileprivate func didTapButton(sender: UIButton) {
      if let delegate = delegate {
         delegate.inputViewDidTapButton(button: sender)
      }
   }
}

extension SBMessageInputView: UITextViewDelegate {

   func textViewShouldBeginEditing(_ textView: UITextView) -> Bool {
      if let delegate = delegate {
         return delegate.inputViewShouldBeginEditing(textView: textView)
```

```
        }
        return true
    }

    func textViewDidBeginEditing(_ textView: UITextView) {

        if (textView.text == placeholder && textView.textColor == placeholderColor) {
            textView.text = ""
            textView.textColor = textColor
        }

        if let delegate = delegate {
            delegate.inputViewDidBeginEditing(textView: textView)
        }
    }

    func textViewDidEndEditing(_ textView: UITextView) {
        if (textView.text.isEmpty) {
            textView.text = placeholder
            textView.textColor = placeholderColor
        }
    }

    func textView(_ textView: UITextView, shouldChangeTextIn range: NSRange, replacementText
text: String) -> Bool {

        if let delegate = delegate {
            return delegate.inputView(textView: textView, shouldChangeTextInRange: range, replace-
mentText: text)
        }
        return true
    }

    func textViewDidChange(_ textView: UITextView) {
        guard let font = textView.font else { return }
        let fontLineHeight = font.lineHeight

        let lines = round((textView.contentSize.height - textView.textContainerInset.top -
textView.textContainerInset.bottom) / fontLineHeight)
        lineHeight = fontLineHeight
        numberOfLines = lines

        if let delegate = delegate {
            delegate.inputViewDidChange(textView: textView)
        }
    }
}
import Foundation
import UIKit

extension UIView {
```

```
    func rotate(_ toValue: CGFloat, duration: CFTimeInterval = 0.2) {
        let animation = CABasicAnimation(keyPath: "transform.rotation")

        animation.toValue = toValue
        animation.duration = duration
        animation.isRemovedOnCompletion = false
        animation.fillMode = CAMediaTimingFillMode.forwards

        self.layer.add(animation, forKey: nil)
    }

}
}
            })
        }
    }
}
```

ДОДАТОК Б. Пристрої які підтримують додаток на ОС iOS

Таблиця Б.1 – Пристрої з підтримкою додатку

Виробник	**Модель**	**Примітки**
Apple	iPhone 5s iPhone 6 iPhone 6 Plus iPhone 6s iPhone 6s Plus iPhone SE iPhone 7 iPhone 7 Plus iPhone 8 iPhone 8 Plus iPhone X iPhone Xs iPhone Xs Max iPhone XR iPhone 11 iPhone 11 Pro iPhone 11 Pro Max	

ДОДАТОК В. Порядок та методики випробувань

1. Тестування перевірки необхідних вимог для виконання

Об'єкт випробувань – ipa-файл, який був встановлений на призстрійй з операційною iOS 13.1.

Мета випробувань – перевірка коректної роботи інтерфейсу для різних типів екранів та роздільних здатностей пристрою.

Тестування проводилось на пристроях iPhone 8 та iPhone 8 Plus під керуванням операційної системи iOS 13.1.

Методика випробувань.

1) ipa-файл встановлюється на обох пристроях.

2) По завершенню встановлення запускається додаток.

3) Пристрій повинен відобразити вікно логіну.

Очікуваний результат:

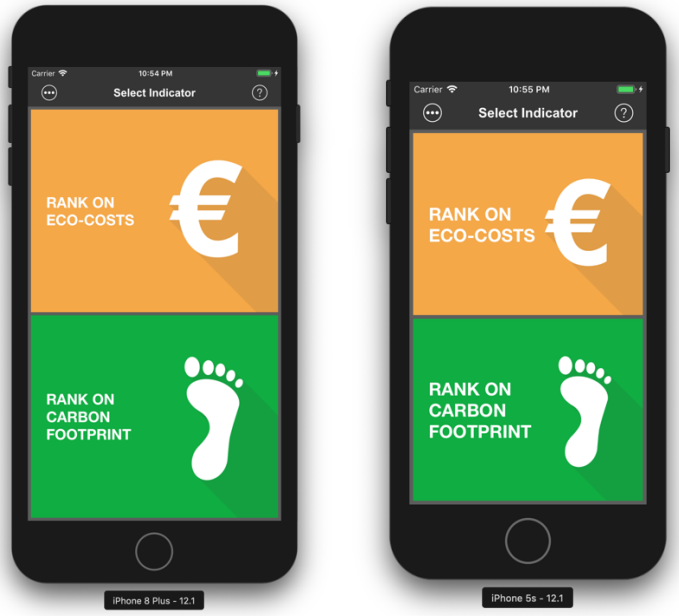

Рисунок В.1 – Вікно логіну

4) Після ведення логіна та пароля користувача перекидає на головний екран :

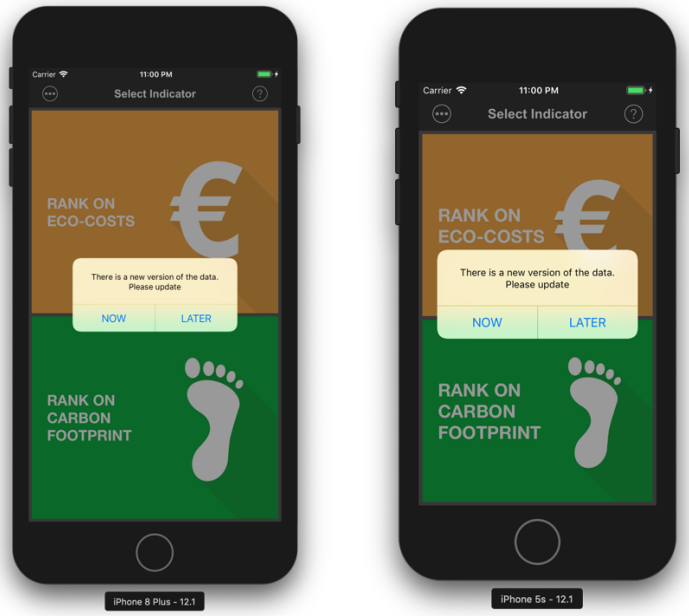

Рисунок В.2 – Вікно головного екрану

5) Після успішного встановлення останньої версії бази даних матеріалів, користувач може вільно навігуватись по додатку.

Результати випробувань

Тест вважається пройденим, якщо виконуються наступні вимоги:

– додаток не закрився с помилкою.

– коректно працює з сервером.

– користувацький інтерфейс не містить помилок.

2. Тестування HR Hub

Об'єкт випробувань – ipa-файл, що встановлюється на пристрій під керуванням ОС iOS 13.1.

Мета випробувань – перевірка наявності необхідних дозволів та наявність коректної роботи додатку.

Засоби і порядок випробувань.

Тестування проводилось на iOS симуляторі iPhone 8 з встановленою операційною системою iOS 13.1.

Методика випробувань.

1) Логін в додаток:

Рисунок В.3 – Логін в додаток

2) Заповнюємо поля логіну і пароля:

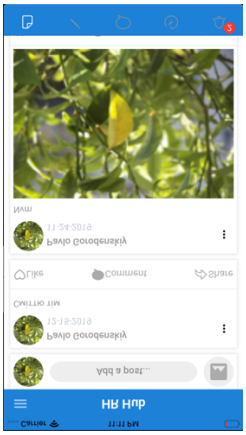

Рисунок В.4 – Результат Логіну в додаток

3) Починаю переходити між екранами:

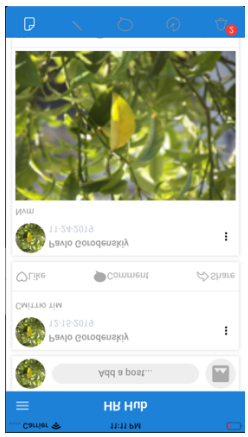

Рисунок В.5 – Головний екран

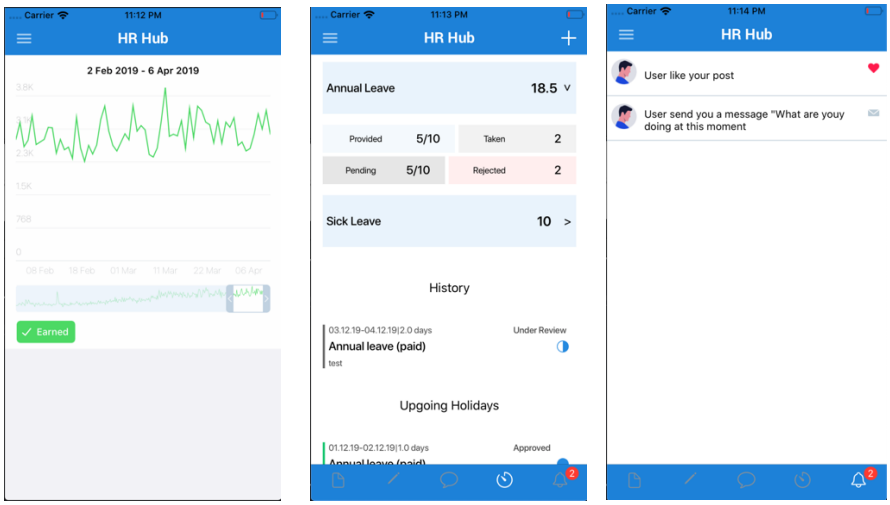

Рисунок В.6 – Екран вихідних

Результати випробувань

Тест вважається пройденим, якщо виконуються наступні вимоги:

– додаток не закрився с помилкою.

– додаток коректно підгрузив всі данні.

– користувацький інтерфейс не містить помилок.