BỘ NÔNG NGHIỆP VÀ PHÁT TRIỂN NÔNG THÔN

**TRƯỜNG ĐẠI HỌC THỦY LỢI**

A blue and white logo

Description automatically generated

**BÀI TẬP LỚN**

**HỌC PHẦN: PHÁT TRIỂN ỨNG DỤNG CHO CÁC THIẾT BỊ DI ĐỘNG**

**ĐỀ TÀI: XÂY DỰNG APP LỊCH**

|  |  |  |  |
| --- | --- | --- | --- |
| Mã Sinh Viên | Họ và Tên | Ngày Sinh | Lớp |
| 2151173783 | Nguyễn Chấn Hưng | 09/12/2003 | 63KTPM1 |
| 2151173748 | Phùng Thế Anh | 01/11/2003 | 63KTPM1 |
| 2151170569 | Nguyễn Duy Chính | 27/11/2003 | 63KTPM1 |
| 2151170562 | Phạm Văn Đăng | 12/10/2002 | 63KTPM1 |

### 

**Hà Nội, năm 2024**

BỘ NÔNG NGHIỆP VÀ PHÁT TRIỂN NÔNG THÔN

**TRƯỜNG ĐẠI HỌC THỦY LỢI**

A blue and white logo

Description automatically generated

**BÀI TẬP LỚN**

**HỌC PHẦN: PHÁT TRIỂN ỨNG DỤNG CHO CÁC THIẾT BỊ DI ĐỘNG**

**ĐỀ TÀI: XÂY DỰNG APP LỊCH**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mã Sinh Viên | Họ và Tên | Ngày Sinh | Điểm | |
| Bằng Số | Bằng Chữ |
| 2151173783 | Nguyễn Chấn Hưng | 09/12/2003 |  |  |
| 2151173748 | Phùng Thế Anh | 01/11/2003 |  |  |
| 2151170569 | Nguyễn Duy Chính | 27/11/2003 |  |  |
| 2151170562 | Phạm Văn Đăng | 12/10/2002 |  |  |

### 

### CÁN BỘ CHẤM THI 1 CÁN BỘ CHẤM THI 2

**Hà Nội, năm 2024**

**LỜI NÓI ĐẦU**

Trong bối cảnh công nghệ phát triển mạnh mẽ, ứng dụng quản lý thời gian trở thành công cụ thiết yếu. Ứng dụng lịch được xây dựng nhằm giúp người dùng dễ dàng quản lý thời gian, công việc, với các tính năng như tạo sự kiện, nhắc nhở, đồng bộ hóa và quản lý công việc theo lịch ngày, tuần, tháng.

Chúng tôi đã thiết kế ứng dụng với giao diện thân thiện và tính năng tối ưu, mong muốn giúp người dùng nâng cao hiệu quả trong cuộc sống. Ứng dụng vẫn sẽ tiếp tục được cải tiến để đáp ứng tốt hơn nhu cầu của người dùng.

**MỤC LỤC**

Lời nói đầu……………………………………………………………………………….3

Phân chia công việc……………………………………………………………………...5

CHƯƠNG 1. MÔ TẢ BÀI TOÁN ………………………………………………………6

1.1. Giới thiệu……………………………………………………………………………7

1.2. Chức năng chính…………………………………………………………………….7

1.3. Yêu cầu phi chức năng………………………………………………………………7

CHƯƠNG 2. PHÂN TÍCH YÊU CẦU VÀ THIẾT KẾ HỆ THỐNG…………………...8

2.1. Phân tích yêu cầu ……………………………………………………………………8

2.2. Thiết kế hệ thống……………………………………………………………………10

2.3. Triển khai……………………………………………………………………………30

2.4. Vận hành và bảo trì………………………………………………………………….35

CHƯƠNG 3. KẾT QUẢ THỰC HIỆN………………………………………………….37

3.1. Công nghệ đã sử dụng……………………………………………………………….37

3.2. Tiến độ thực hiện…………………………………………………………………….37

3.3. Hình ảnh sản phẩm…………………………………………………………………..37

Kết luận………………………………………………………………...…………………40

Phụ lục…………………………………………………………………………………….42

**Phân chia công việc:**

|  |  |
| --- | --- |
| Nguyễn Chấn Hưng | Figma, sơ đồ lớp, chức năng giao diện báo thức, xem thời tiết, xem map |
| Phùng Thế Anh | Đăng ký, Đăng nhập, Đăng xuất, Quên mật khẩu, Xem thời gian các nước khác |
| Nguyễn Duy Chính | Hiển thị lịch, thêm lịch trình, xem lịch âm dương, bấm giờ, danh sách việc cần làm |
| Phạm Văn Đăng | Viết báo cáo, xem ngày lễ, quản lý tài khoản |

**CHƯƠNG 1. MÔ TẢ BÀI TOÁN**

**1.1. Giới thiệu**

Ứng dụng lịch này cung cấp các tính năng quản lý sự kiện, thông báo, và nhắc nhở cho người dùng, được xây dựng trên nền tảng Android với nhiều hoạt động hỗ trợ như quản lý sự kiện, thông báo, đồng hồ báo thức, và tương tác với người dùng qua đăng ký và đăng nhập.

**1.2. Chức năng chính**

Ứng dụng Calendar cần có các chức năng sau:

Quản lý sự kiện và nhiệm vụ: Người dùng có thể tạo, chỉnh sửa, và xem các sự kiện lịch và nhiệm vụ.

Báo thức và thông báo: Hỗ trợ nhắc nhở bằng báo thức và thông báo đẩy cho các sự kiện hoặc nhiệm vụ.

Hiển thị thời tiết: Cập nhật thông tin thời tiết liên quan đến vị trí của người dùng hoặc sự kiện.

Bản đồ: Hiển thị vị trí của người dùng

Đồng hồ đếm ngược: Giúp người dùng theo dõi thời gian hoàn thành nhiệm vụ.

Quản lý người dùng: Đăng nhập, đăng ký và quản lý hồ sơ cá nhân.

**1.3. Yêu cầu phi chức năng**

* Thời gian phản hồi: Ứng dụng phải có thời gian phản hồi dưới 2 giây cho các hành động người dùng, chẳng hạn như tạo, chỉnh sửa hoặc xóa sự kiện.
* Tối ưu hóa tài nguyên: Ứng dụng cần sử dụng hiệu quả CPU và bộ nhớ để không làm chậm thiết bị, đặc biệt khi chạy trong nền.
* Tính ổn định: Ứng dụng phải hoạt động ổn định mà không gặp phải các lỗi nghiêm trọng hoặc sự cố. Tỷ lệ lỗi phải thấp, dưới 1%.
* Khôi phục sau sự cố: Nếu ứng dụng bị treo hoặc gặp sự cố, nó cần có khả năng phục hồi dữ liệu trước đó mà không bị mất thông tin của người dùng.
* Hỗ trợ nhiều người dùng: Ứng dụng cần có khả năng xử lý hàng triệu người dùng mà không làm giảm hiệu suất, đặc biệt trong các hoạt động như gửi thông báo hoặc đồng bộ hóa dữ liệu.
* Khả năng mở rộng chức năng: Kiến trúc ứng dụng phải cho phép bổ sung các tính năng mới mà không cần phải tái cấu trúc toàn bộ.
* Bảo mật dữ liệu người dùng: Tất cả dữ liệu cá nhân và sự kiện của người dùng cần được mã hóa khi lưu trữ và truyền tải.
* Giao diện người dùng thân thiện: Giao diện phải dễ sử dụng và trực quan, với khả năng truy cập nhanh đến các chức năng chính.
* Hỗ trợ nhiều thiết bị: Ứng dụng cần hoạt động trên nhiều loại thiết bị Android khác nhau, từ smartphone đến tablet.
* Tương thích với các phiên bản Android: Ứng dụng phải tương thích với các phiên bản Android phổ biến, chẳng hạn như Android 8.0 (Oreo) trở lên.
* Dễ bảo trì và cập nhật: Codebase của ứng dụng phải được tổ chức tốt và dễ dàng bảo trì, với các quy trình rõ ràng cho việc cập nhật và phát triển thêm tính năng mới.
* Tùy chỉnh giao diện: Cho phép người dùng tùy chỉnh giao diện (chủ đề, màu sắc) theo sở thích cá nhân.

**CHƯƠNG 2. PHÂN TÍCH YÊU CẦU VÀ THIẾT KẾ HỆ THỐNG**

**2.1. Phân tích yêu cầu:**

**Xác định người dùng:**

* Người dùng cuối: Là những cá nhân sử dụng ứng dụng lịch để quản lý công việc cá nhân và các sự kiện hàng ngày. Nhóm người dùng có thể bao gồm:
* Học sinh, sinh viên cần theo dõi lịch học, thời hạn bài tập, và các hoạt động ngoại khóa.
* Nhân viên văn phòng quản lý các cuộc họp, sự kiện công việc, và công việc cá nhân.
* Bất kỳ ai cần lập kế hoạch cho các hoạt động trong cuộc sống hàng ngày (ví dụ: sinh nhật, cuộc hẹn, sự kiện gia đình).
* Người dùng có các mức độ hiểu biết về công nghệ khác nhau, từ những người không quen thuộc với ứng dụng di động đến những người thành thạo. Vì vậy, ứng dụng cần đảm bảo tính đơn giản, dễ tiếp cận với người dùng.

**Thu thập yêu cầu:**

* Tạo mới, chỉnh sửa, xóa và hiển thị các sự kiện hoặc nhiệm vụ.
* Thiết lập báo thức hoặc nhắc nhở cho các sự kiện và nhiệm vụ.
* Xem danh sách sự kiện và nhiệm vụ dưới nhiều dạng (theo ngày, tuần, tháng).
* Tích hợp thông tin thời tiết để hỗ trợ người dùng lập kế hoạch cho các sự kiện.
* Hiển thị vị trí sự kiện trên bản đồ và cung cấp hướng dẫn đường đi.
* Đăng nhập, đăng ký tài khoản để quản lý dữ liệu cá nhân và đồng bộ dữ liệu trên nhiều thiết bị.
* Cung cấp tùy chọn cho người dùng để nhận thông báo đẩy về sự kiện và nhiệm vụ sắp tới.
* Cho phép người dùng tùy chỉnh giao diện và cài đặt cá nhân hóa cho ứng dụng.

**Phân tích yêu cầu:**

Quản lý sự kiện và nhiệm vụ:

* Tạo mới, chỉnh sửa, xóa sự kiện/nhiệm vụ: Người dùng có thể thêm, sửa đổi hoặc xóa sự kiện và nhiệm vụ. Điều này bao gồm việc nhập thông tin chi tiết như tên, thời gian, địa điểm và mô tả.
* Hiển thị sự kiện/nhiệm vụ: Cung cấp giao diện để người dùng xem các sự kiện và nhiệm vụ theo nhiều dạng khác nhau như lịch tháng, tuần, ngày, giúp họ dễ dàng quản lý thời gian.

Báo thức và nhắc nhở:

* Thiết lập báo thức: Cho phép người dùng đặt báo thức cho sự kiện hoặc nhiệm vụ, bao gồm tùy chọn âm thanh và tần suất báo thức.
* Thông báo đẩy: Gửi thông báo đến thiết bị của người dùng để nhắc nhở họ về các sự kiện hoặc nhiệm vụ sắp tới.

Thông tin thời tiết:

* Tích hợp thời tiết: Cung cấp thông tin thời tiết theo thời gian thực để người dùng có thể lên kế hoạch cho các sự kiện của họ, bao gồm nhiệt độ, điều kiện thời tiết, và dự báo cho các ngày tới.

Quản lý người dùng:

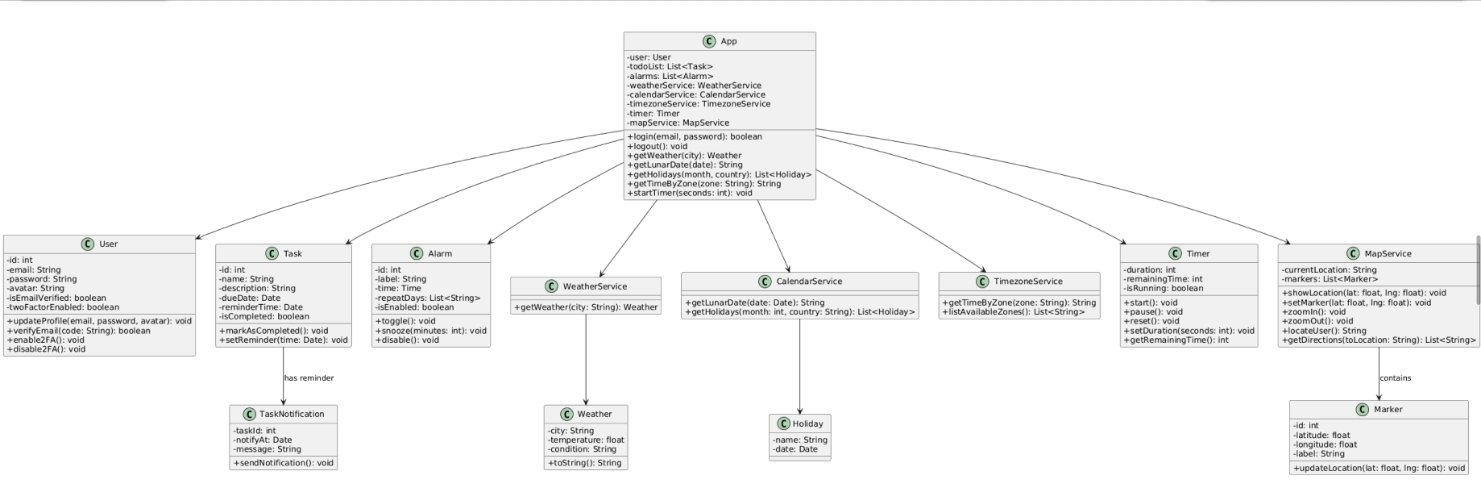
* Đăng nhập và đăng ký: Cho phép người dùng tạo tài khoản mới và đăng nhập vào ứng dụng để quản lý dữ liệu cá nhân.
* Đồng bộ hóa dữ liệu: Đảm bảo rằng dữ liệu cá nhân (sự kiện, nhiệm vụ) được đồng bộ hóa trên nhiều thiết bị mà người dùng sử dụng.

Tìm kiếm và điều hướng:

* Tìm kiếm sự kiện/nhiệm vụ: Cung cấp tính năng tìm kiếm để người dùng nhanh chóng tìm thấy sự kiện hoặc nhiệm vụ theo tên hoặc ngày tháng.
* Tìm kiếm địa điểm trên bản đồ: Cho phép người dùng tìm kiếm địa điểm cho sự kiện và hiển thị nó trên bản đồ.

**2.2. Thiết kế hệ thống:**

**Biểu đồ lớp cho mô hình miền:**



**Biểu đồ lớp cho MVC:**

**Thiết kế kiến trúc (MVC):**

**1. Model (Mô hình dữ liệu):**

Event: Lưu trữ thông tin về sự kiện lịch.

Task: Lưu trữ thông tin về các nhiệm vụ cần thực hiện.

WeatherResponse: nhiệt độ, độ ẩm, điều kiện thời tiết, dữ liệu thời tiết được trả về từ API.

TaskDao và TaskDaoImpl: Quản lý dữ liệu nhiệm vụ trong cơ sở dữ liệu.

UserDao và UserDaoImpl: Quản lý thông tin người dùng (đăng nhập, đăng ký).

**2. View (Giao diện hiển thị):**

CalendarFragment: Hiển thị lịch với các sự kiện và nhiệm vụ.

WeatherFragment: Hiển thị thông tin thời tiết cho người dùng.

TaskFragment: Hiển thị danh sách nhiệm vụ.

ShowTaskActivity: Hiển thị thông tin chi tiết của một nhiệm vụ cụ thể.

Login, MainSignUp, Sign\_Up, UserProfile: Các màn hình cho việc quản lý tài khoản người dùng.

NotificationActivity: Hiển thị các thông báo cho người dùng.

EventAdapter: Quản lý hiển thị danh sách sự kiện.

TaskAdapter: Quản lý hiển thị danh sách nhiệm vụ.

**3. Controller (Bộ điều khiển):**

AlarmActivity: Quản lý các chức năng liên quan đến báo thức, xử lý sự kiện từ người dùng để thêm, sửa hoặc xóa báo thức.

CreateTaskActivity: Nhận dữ liệu từ người dùng để tạo nhiệm vụ mới và cập nhật mô hình.

UpdateTaskActivity: Xử lý việc chỉnh sửa nhiệm vụ hiện có.

NotificationHelper: Tạo và quản lý thông báo cho sự kiện và nhiệm vụ.

NotificationReceiver: Nhận và xử lý các thông báo từ hệ thống.

MapActivity: Quản lý việc hiển thị bản đồ và tìm kiếm địa điểm cho sự kiện.

TimerActivity: Quản lý các chức năng liên quan đến đồng hồ đếm ngược.

MainActivity: Giao diện chính tổng hợp các chức năng và chuyển đổi giữa các phần của ứng dụng.

**Dựa trên kiến trúc MVC đã chọn, ta có thể xác định các lớp sau:**

* **Lớp Event**
  + **Thuộc tính:**
* String date: ngày diễn ra sự kiện
* String description: mô tả chi tiết của sự kiện.
  + **Phương thức:**
    - public Event(String date, String description): hàm tạo, khởi tạo đối tượng với ngày và mô tả sự kiện.
    - public String getDate(): trả về ngày của sự kiện.
    - public String getDescription(): trả về mô tả của sự kiện.
* **Lớp Task**
  + **Thuộc tính:**
    - int id: mã định danh duy nhất của công việc.
    - String taskName: tên của công việc.
    - String taskDescription: mô tả chi tiết về công việc.
    - String date: ngày diễn ra công việc.
    - String startTime: giờ bắt đầu công việc.
    - String endTime: giờ kết thúc công việc.
    - String alarmTime: giờ báo thức cho công việc.
    - String color: màu sắc liên quan đến công việc (có thể dùng để phân loại).
    - String location: vị trí hoặc địa điểm của công việc.
    - String idCurrentUser: ID của người dùng hiện tại liên kết với công việc.
  + **Phương thức:**
* public Task(): khởi tạo đối tượng Task mặc định.
* public Task(int id, String taskName, String taskDescription, String date, String startTime, String endTime, String alarmTime, String color, String location, String idCurrentUser): khởi tạo đối tượng Task với các thuộc tính được truyền vào.

public int getId(): trả về ID của nhiệm vụ.

* public void setId(int id): thiết lập ID cho nhiệm vụ.
* public String getTaskName(): trả về tên nhiệm vụ.
* public void setTaskName(String taskName): thiết lập tên cho nhiệm vụ.
* public String getTaskDescription(): trả về mô tả nhiệm vụ.
* public void setTaskDescription(String taskDescription): thiết lập mô tả cho nhiệm vụ.
* public String getDate(): trả về ngày diễn ra nhiệm vụ.
* public void setDate(String date): thiết lập ngày cho nhiệm vụ.
* public String getStartTime(): trả về thời gian bắt đầu của nhiệm vụ.
* public void setStartTime(String startTime): thiết lập thời gian bắt đầu cho nhiệm vụ.
* public String getEndTime(): trả về thời gian kết thúc của nhiệm vụ.
* public void setEndTime(String endTime): thiết lập thời gian kết thúc cho nhiệm vụ.
* public String getAlarmTime(): trả về thời gian báo thức của nhiệm vụ.
* public void setAlarmTime(String alarmTime): thiết lập thời gian báo thức cho nhiệm vụ.
* public String getColor(): trả về màu sắc của nhiệm vụ.
* public void setColor(String color): thiết lập màu sắc cho nhiệm vụ.
* public String getLocation(): trả về địa điểm của nhiệm vụ.
* public void setLocation(String location): thiết lập địa điểm cho nhiệm vụ.
* public String getIdCurrentUser(): trả về ID của người dùng hiện tại.
* public void setIdCurrentUser(String idCurrentUser): thiết lập ID của người dùng cho nhiệm vụ.
* **Lớp WeatherResponse**
  + **Thuộc tính:**
    - String cityName: tên thành phố nơi lấy thông tin thời tiết.
    - String temperature: nhiệt độ hiện tại của thành phố.
    - String description: mô tả ngắn về điều kiện thời tiết (ví dụ: "mây rải rác", "nắng").
    - String icon: mã icon đại diện cho điều kiện thời tiết (được sử dụng để hiển thị hình ảnh thời tiết).
  + **Phương thức:**
    - public WeatherResponse(String cityName, String temperature, String description, String icon): khởi tạo đối tượng WeatherResponse với các thông tin về thời tiết.
    - public String getCityName(): trả về tên thành phố.
    - public String getTemperature(): trả về nhiệt độ hiện tại.
    - public String getDescription(): trả về mô tả điều kiện thời tiết.
    - public String getIcon(): trả về mã icon thời tiết.
* **Lớp TaskDaoImpl**
  + **Thuộc tính:**
    - FirebaseFirestore db: đối tượng dùng để truy xuất và thao tác với cơ sở dữ liệu Firestore.
    - CollectionReference tasksCollection: tham chiếu đến tập hợp dữ liệu "Tasks" trong Firestore.
    - static TaskDaoImpl instance: một thể hiện duy nhất của lớp TaskDaoImpl
  + **Phương thức:**
* TaskDaoImpl(): Constructor khởi tạo một đối tượng TaskDaoImpl, thiết lập kết nối với Firestore và tham chiếu đến bộ sưu tập "Tasks.
* static TaskDaoImpl getInstance(): Phương thức tĩnh để lấy thể hiện đơn của TaskDaoImpl, đảm bảo chỉ có một thể hiện duy nhất trong suốt vòng đời ứng dụng.
* void getAllTasks(String idUser, final getAllTasksCallBack callback): Lấy tất cả các nhiệm vụ (tasks) của người dùng với ID cụ thể và trả về danh sách thông qua callback.
* void getTaskOfDay(String date, String idUser, final FirebaseCallback callback): Lấy các nhiệm vụ cho một ngày cụ thể của người dùng và trả về danh sách thông qua callback.
* void createTask(Task task): Tạo một nhiệm vụ mới trong Firestore với các thông tin được cung cấp trong đối tượng Task.
* void updateTask(int taskId, Task task): Cập nhật thông tin của một nhiệm vụ trong Firestore dựa trên ID của nó.
* void deleteTask(int taskId): Xóa một nhiệm vụ khỏi Firestore dựa trên ID của nó.
* void fetchLastTaskId(final LastTaskIdCallback callback): Lấy ID của nhiệm vụ cuối cùng được thêm vào bộ sưu tập và trả về qua callback.
* void isTaskExists(Task task, final TaskExistsCallback callback): Kiểm tra xem một nhiệm vụ có tồn tại trong Firestore hay không và trả về nhiệm vụ nếu tồn tại hoặc null nếu không.
* **Lớp UserDaoImpl**
  + **Thuộc tính:**
    - FirebaseAuth firebaseAuth: đối tượng dùng để quản lý xác thực người dùng thông qua Firebase Authentication.
    - static UserDaoImpl instance: một thể hiện duy nhất của lớp UserDaoImpl (singleton pattern).
  + **Phương thức:**
* private UserDaoImpl(): khởi tạo đối tượng UserDaoImpl và khởi tạo firebaseAuth với thể hiện của FirebaseAuth.
* public static UserDaoImpl getInstance(): trả về thể hiện duy nhất của UserDaoImpl. Phương thức này đảm bảo rằng chỉ có một đối tượng duy nhất được tạo ra cho lớp này.
* getIdCurrentUser(): trả về ID của người dùng hiện tại (UID). Nếu không có người dùng nào đang đăng nhập, phương thức này sẽ trả về "0".
* **Lớp CalendarFragment**
  + **Thuộc tính:**
    - MaterialCalendarView calendar: Thành phần giao diện để hiển thị lịch.
    - CalendarDay selectedDate: Ngày hiện tại đang được chọn trong lịch.
    - TextView tvDuong: Hiển thị ngày dương lịch.
    - TextView tvAm: Hiển thị ngày âm lịch.
    - RecyclerView recyclerView: Thành phần giao diện để hiển thị danh sách sự kiện.
    - EventAdapter eventAdapter: Adapter cho RecyclerView, dùng để hiển thị danh sách sự kiện.
    - List<Event> eventList: Danh sách chứa tất cả các sự kiện.
    - List<Event> filteredEventList: Danh sách chứa các sự kiện đã lọc dựa trên tháng đã chọn.
  + **Phương thức:**
    - CalendarFragment(): Constructor của lớp.
    - onCreateView: Phương thức khởi tạo giao diện cho fragment, thiết lập các thành phần giao diện và sự kiện.
    - updateDateTextViews(CalendarDay date): Cập nhật các TextView hiển thị ngày dương và âm lịch dựa trên ngày đã chọn.
    - convertSolarToLunar(CalendarDay date): Chuyển đổi ngày dương lịch sang ngày âm lịch.
    - loadEventData(): Tải danh sách các sự kiện.
    - updateEventListForSelectedMonth(CalendarDay date): Cập nhật danh sách sự kiện dựa trên tháng của ngày đã chọn.
    - highlightDates(List<Event> eventList): Nổi bật các ngày có sự kiện trong lịch.
* **Lớp WeatherFragment**
  + **Thuộc tính:**
    - **TextView tvCityName**: Hiển thị tên thành phố.
    - **TextView tvTemperature**: Hiển thị nhiệt độ hiện tại.
    - **TextView tvWeatherDescription**: Hiển thị mô tả thời tiết.
    - **ImageView ivWeatherIcon**: Hiển thị biểu tượng thời tiết.
    - **ProgressBar progressBar**: Thanh tiến trình để chỉ ra việc tải dữ liệu.
    - **Spinner spinnerCities**: Thành phần cho phép người dùng chọn thành phố.
    - **static final String API\_KEY**: Khóa API cho dịch vụ thời tiết.
    - **static final String UNITS**: Đơn vị đo nhiệt độ (metric cho độ C).
    - **String[] cities**: Danh sách các thành phố có sẵn để chọn.
  + Phương thức:
    - WeatherFragment(): Constructor của lớp.
    - onCreateView: Phương thức khởi tạo giao diện cho fragment.
    - fetchWeatherData(String cityName): Lấy dữ liệu thời tiết cho thành phố đã chọn và cập nhật giao diện.
    - parseWeatherResponse(String response): Phân tích dữ liệu thời tiết từ phản hồi JSON để tạo ra đối tượng WeatherResponse.
* **Lớp TaskFragment**
  + **Thuộc tính:**
    - ListTask: Danh sách các nhiệm vụ hiện có.
    - TaskAdapter taskAdapter: Bộ điều hợp cho RecyclerView để hiển thị danh sách nhiệm vụ.
    - RecyclerView rcvTask: Thành phần giao diện để hiển thị danh sách nhiệm vụ.
    - MaterialCalendarView calendar: Thành phần giao diện cho lịch để chọn ngày.
    - String dateSelected: Ngày đã chọn.
    - String idCurrentUser: ID của người dùng hiện tại.
    - Button btnCreateTask: Nút để tạo nhiệm vụ mới.
    - CalendarDay selectedDate: Ngày hiện tại được chọn trên lịch.
    - ImageView gifImageView: Thành phần để hiển thị GIF khi không có nhiệm vụ.
  + **Phương thức:**
    - askFragment(): Constructor của lớp.
    - onCreateView: Phương thức khởi tạo giao diện của fragment.
    - initUi(View rootView): Thiết lập các thành phần giao diện trong Fragment.
    - getDate(): Lấy ngày hiện tại hoặc từ Intent.
    - highlightDates(List<Task> mListTask): Đánh dấu các ngày có nhiệm vụ trên lịch.
    - changeDay(): Thiết lập sự kiện khi người dùng thay đổi ngày trên lịch.
    - getListTasksOfDay(String dateSelected, String idCurrentUser): Lấy danh sách nhiệm vụ cho một ngày cụ thể từ cơ sở dữ liệu.
    - clickShowItem(Task task): Mở ShowTaskActivity để hiển thị chi tiết nhiệm vụ đã chọn.
    - clickUpdateItem(Task task): Mở UpdateTaskActivity để cập nhật nhiệm vụ đã chọn.
    - deleteTask(Task task): Xóa nhiệm vụ đã chọn sau khi xác nhận từ người dùng.
* **Lớp ShowTaskActivity:**
  + **Thuộc tính:**
* TextView show\_taskName: Hiển thị tên nhiệm vụ.
* TextView show\_startTime: Hiển thị thời gian bắt đầu của nhiệm vụ.
* TextView show\_endTime: Hiển thị thời gian kết thúc của nhiệm vụ.
* TextView show\_date: Hiển thị ngày của nhiệm vụ.
* TextView show\_alarmTime: Hiển thị thời gian báo thức của nhiệm vụ.
* TextView show\_taskDescription: Hiển thị mô tả của nhiệm vụ.
* TextView show\_location: Hiển thị địa điểm của nhiệm vụ.
* TextView color: Hiển thị màu sắc của nhiệm vụ.
* ImageView show\_imageColor: Hiển thị hình ảnh biểu tượng màu sắc tương ứng với nhiệm vụ.
* ImageButton img\_buttonback: Nút quay lại để trở về màn hình trước.
* int id: Lưu trữ ID của nhiệm vụ đang hiển thị.
  + **Phương thức:**
* onCreate(Bundle savedInstanceState): Phương thức khởi tạo cho Activity, nơi thiết lập giao diện người dùng và thực hiện các bước khởi tạo ban đầu.
* initUi() để khởi tạo các thành phần giao diện.
* getTask() để lấy thông tin nhiệm vụ từ Intent.
* setupBackButton() để thiết lập hành động cho nút quay lại.
* public void initUi(): Phương thức để ánh xạ các thành phần giao diện từ XML với các biến tương ứng trong lớp.
* public void getTask(): Phương thức để lấy thông tin nhiệm vụ từ Intent và cập nhật giao diện với thông tin nhiệm vụ.
* private void setupBackButton(): Phương thức để thiết lập hành động cho nút quay lại. Khi nút được nhấn, nó sẽ gọi onBackPressed() để trở về màn hình trước
* **Lớp Login, MainSignUp, Sign\_Up, UserProfile**

**Login:**

* + **Thuộc tính:**
    - edt\_Email: Trường nhập email của người dùng.
    - edt\_Password: Trường nhập mật khẩu của người dùng.
    - tv\_Email: Hiển thị nhãn cho trường nhập email.
    - tv\_Password: Hiển thị nhãn cho trường nhập mật khẩu.
    - tv\_ForPassword: Liên kết cho việc quên mật khẩu.
    - tv\_messes: Thông báo lỗi cho trường email.
    - tv\_messes1: Thông báo lỗi cho trường mật khẩu.
    - sw\_Remmember: Công tắc để nhớ đăng nhập.
    - imgbtn\_Facebook: Nút đăng nhập bằng Facebook.
    - imbtn\_Google: Nút đăng nhập bằng Google.
    - imgbtn\_Twitter: Nút đăng nhập bằng Twitter.
    - showPasswork: Nút để hiển thị/ẩn mật khẩu.
    - back: Nút quay lại trang đăng ký.
    - btn\_Login: Nút để thực hiện đăng nhập.
    - ll\_Email: Layout cho trường nhập email.
    - ll\_passWord: Layout cho trường nhập mật khẩu.
    - firebaseAuth: Đối tượng để quản lý xác thực người dùng Firebase.
    - firebaseFirestore: Đối tượng để quản lý cơ sở dữ liệu Firestore.
    - googleSignInClient: Đối tượng để quản lý xác thực Google.
    - progressDialog: Đối tượng để hiển thị tiến trình trong khi tạo tài khoản.
* RC\_SIGN\_IN: Hằng số cho mã yêu cầu đăng nhập Google.
* isPasswordVisible: Biến boolean để kiểm soát việc hiển thị mật khẩu.
  + **Phương thức:**
* **mapping()**: Phương thức dùng để ánh xạ các thành phần giao diện (View) từ layout vào các thuộc tính.
* **onCreate(Bundle savedInstanceState)**: Phương thức khởi tạo, nơi thiết lập giao diện người dùng, khởi tạo Firebase, và đăng ký các sự kiện cho các nút bấm.
* **signIn()**: Phương thức khởi động quy trình đăng nhập Google.
* **onActivityResult(int requestCode, int resultCode, @Nullable Intent data)**: Phương thức xử lý kết quả từ các hoạt động con, như đăng nhập Google.
* **firebaseAuth(String idToken)**: Phương thức xác thực người dùng với Firebase sử dụng mã thông báo ID từ Google.
* **showPasswork ClickListener**: Thay đổi chế độ hiển thị mật khẩu giữa ẩn và hiện khi người dùng nhấn nút hiển thị mật khẩu.
* **tv\_ForPassword ClickListener**: Mở hộp thoại để đặt lại mật khẩu khi người dùng nhấn vào liên kết quên mật khẩu.
* **btn\_Login ClickListener**: Xử lý sự kiện khi người dùng nhấn nút đăng nhập, thực hiện xác thực với Firebase.

**MainSignUp:**

* + **Thuộc tính:**
    - btn\_PhoneNumber: Nút để đăng ký bằng số điện thoại.
    - btn\_Facebook: Nút để đăng ký bằng tài khoản Facebook.
    - btn\_Google: Nút để đăng ký bằng tài khoản Google.
    - btn\_Login: Nút để chuyển đến màn hình đăng nhập.
    - **Firebase:**firebaseAuth: Đối tượng FirebaseAuth để xử lý xác thực người dùng.
    - firebaseFirestore: Đối tượng FirebaseFirestore để xử lý cơ sở dữ liệu.
    - storageReference: Đối tượng StorageReference để xử lý lưu trữ hình ảnh trên Firebase.
    - progressDialog: Đối tượng ProgressDialog để hiển thị trạng thái trong quá trình tạo tài khoản.
    - googleSignInClient: Đối tượng GoogleSignInClient để quản lý đăng nhập Google.
    - mcallbackManager: Đối tượng CallbackManager để quản lý callback từ Facebook.
    - authStateListener: Listener để theo dõi trạng thái xác thực của Firebase.
    - accessTokenTracker: Đối tượng AccessTokenTracker để theo dõi token truy cập từ Facebook.
    - imageView: ShapeableImageView để hiển thị ảnh đại diện người dùng.
  + **Phương thức:**
* private void Mapping(): Phương thức để ánh xạ các view từ layout XML vào các thuộc tính trong lớp.
* protected void onCreate(Bundle savedInstanceState): Phương thức khởi tạo, nơi thiết lập UI và cấu hình các nút sự kiện.
* protected void onStart(): Phương thức được gọi khi Activity bắt đầu, thêm listener cho FirebaseAuth.
* protected void onStop(): Phương thức được gọi khi Activity dừng, loại bỏ listener cho FirebaseAuth.
* private void handleFacebookAccessToken(AccessToken accessToken): Phương thức xử lý đăng nhập Facebook bằng cách lấy credential từ token và xác thực với Firebase.
* private void signIn(): Phương thức để bắt đầu quá trình đăng nhập Google.
* protected void onActivityResult(int requestCode, int resultCode, Intent data): Phương thức xử lý kết quả của activity đăng nhập Google.
* private void firebaseAuth(String idToken): Phương thức xác thực người dùng với Firebase bằng idToken từ Google.
* private void uploadImageToFirebase(Uri photoUrl): Phương thức để tải lên hình ảnh đại diện người dùng từ Google vào Firebase Storage.
* Các Toast thông báo cho người dùng về các hành động thành công hoặc thất bại.

**Sign\_Up:**

* + **Thuộc tính:**
* tv\_Name: Hiển thị tên người dùng.
* tv\_Email: Hiển thị địa chỉ email.
* tv\_Password: Hiển thị mật khẩu.
* messes: Thông báo cho tên người dùng.
* messes1: Thông báo cho email.
* messes2: Thông báo cho mật khẩu.
* edt\_Name: Nhập tên người dùng.
* edt\_Email: Nhập địa chỉ email.
* edt\_Password: Nhập mật khẩu.
* img\_Showpass: Nút hiển thị/ẩn mật khẩu.
* btn\_SignUp: Nút để đăng ký.
* isPasswordVisible: Để xác định trạng thái hiển thị mật khẩu (ẩn/hiện).
* firebaseAuth: Đối tượng FirebaseAuth để xử lý xác thực người dùng.
* firebaseFirestore: Đối tượng FirebaseFirestore để lưu trữ dữ liệu người dùng.
* userID: ID người dùng sau khi đăng ký.
* TAG: Một chuỗi để gán cho thông báo ghi nhật ký.
  + **Phương thức:**
* **Mapping()**:ánh xạ (bind) các thành phần giao diện từ layout với các thuộc tính trong lớp.
* **onCreate(Bundle savedInstanceState)**: Nó thiết lập giao diện người dùng và các thuộc tính ban đầu cho lớp Sign\_Up.
* setOnFocusChangeListener: Để thay đổi màu sắc của nhãn khi trường nhập liệu được chọn.
* setOnClickListener: Để hiển thị/ẩn mật khẩu hoặc thực hiện hành động khi nhấn nút đăng ký.
* **TextWatcher**: theo dõi sự thay đổi của nội dung trong các trường nhập liệu. Nó kiểm tra độ dài của các chuỗi nhập vào và cập nhật thông báo lỗi hoặc thông báo thành công.
* **btn\_SignUp.setOnClickListener**: Lấy thông tin từ các trường nhập liệu.Kiểm tra điều kiện độ dài của email và mật khẩu; Gọi phương thức createUserWithEmailAndPassword để đăng ký người dùng mới với Firebase; Gửi email xác minh cho người dùng mới; Lưu thông tin người dùng vào Firestore; Chuyển hướng người dùng đến MainActivity.

**UserProfile:**

* + **Thuộc tính:**
* tvName: Hiển thị tên người dùng.
* tvEmail: Hiển thị email người dùng.
* tvPassword: Hiển thị mật khẩu người dùng.
* tvmessageVerification: Hiển thị thông báo xác minh email.
* tvresetPasswordYP: Hiển thị nút reset mật khẩu.
* tvaddImageYP: Hiển thị nút thêm ảnh.
* imageUser: Hiển thị ảnh đại diện của người dùng.
* btnlogOut: Nút đăng xuất.
* btnimgback: Nút quay lại (back button).
* FirebaseAuth firebaseAuth: Đối tượng xác thực Firebase.
* FirebaseFirestore firebaseFirestore: Đối tượng Firestore để lưu trữ dữ liệu.
* StorageReference storageReference: Đối tượng lưu trữ ảnh trên Firebase.
* FirebaseUser user: Đối tượng người dùng hiện tại.
* String userID: ID của người dùng hiện tại.
  + **Phương thức:**
* mapping(): Khởi tạo các thuộc tính bằng cách ánh xạ chúng với các thành phần giao diện trong activity\_user\_profile.xml.
* onCreate(Bundle savedInstanceState): Phương thức khởi tạo hoạt động, nơi thiết lập giao diện người dùng, Firebase, và xử lý các sự kiện người dùng.
* onActivityResult(int requestCode, int resultCode, @Nullable Intent data): Xử lý kết quả trả về từ một hoạt động con (như chọn hình ảnh từ thư viện).
* uploadImageToFirebase(Uri imageUri): Tải hình ảnh lên Firebase Storage và lưu URL hình ảnh vào Firestore.
* setContentView(): Thiết lập giao diện của hoạt động.
* Dialog Creation and Handling: Tạo và quản lý hộp thoại để reset mật khẩu của người dùng.
* Firebase Interaction:Gửi email xác minh. Cập nhật mật khẩu người dùng và thông tin người dùng trong Firestore.
* **Lớp NotificationActivity**
  + **Thuộc tính:**
    - NotificationChannel : thông báo dùng để quản lý các thông báo liên quan đến nhiệm vụ.
    - NotificationManager:Đối tượng quản lý thông báo, chịu trách nhiệm tạo và quản lý các thông báo.
  + **Phương thức:**
    - onCreate(Bundle savedInstanceState):Phương thức khởi tạo hoạt động, thiết lập giao diện người dùng từ tệp XML activity\_notifications.xml.
    - EdgeToEdge.enable(this):
    - Cho phép thiết lập giao diện với các tính năng giao diện "edge-to-edge" (nội dung tràn toàn màn hình).
* **Lớp EventAdapter:**
  + **Thuộc tính:**
    - List<Event>: Danh sách các sự kiện (events) được cung cấp cho adapter để hiển thị trong RecyclerView
  + **Phương thức:**
    - EventAdapter(List<Event> eventList) :Khởi tạo đối tượng EventAdapter với danh sách sự kiện được truyền vào qua tham số eventList.
    - onCreateViewHolderPhương thức này được gọi khi cần tạo ra một ViewHolder mới.
    - onBindViewHolder: Gán dữ liệu từ danh sách eventList vào các mục cụ thể trong ViewHolder.
    - getItemCount():Trả về số lượng sự kiện trong eventList.
* **Lớp TaskAdapter**
  + **Thuộc tính:**
* mListTask: Danh sách các công việc (task) được cung cấp cho adapter để hiển thị trong RecyclerView
* iClickListener: Interface cung cấp các phương thức callback để xử lý sự kiện khi người dùng nhấn vào một mục (item) hoặc một menu của mục đó.
  + **Phương thức:**
    - TaskAdapter(List<Task> mListTask, IClickListener iClickListener) (constructor): Khởi tạo đối tượng TaskAdapter với danh sách các công việc (mListTask) và đối tượng lắng nghe sự kiện click (iClickListener).
    - setData(List<Task> list): Cập nhật danh sách các công việc mới và gọi notifyDataSetChanged() để cập nhật giao diện người dùng.
    - onCreateViewHolder: Tạo và trả về một đối tượng TaskViewHolder mới, sử dụng layout item\_task.xml để định dạng cho từng mục trong RecyclerView.
    - onBindViewHolder: Gán dữ liệu từ danh sách mListTask vào các phần tử của ViewHolder tương ứng với vị trí position. Tùy vào giá trị thuộc tính color của công việc, nó đặt biểu tượng màu sắc thích hợp.
    - getItemCount(): Trả về số lượng công việc trong danh sách mListTask
* **Lớp AlarmActivity**
  + **Thuộc tính:**
* timePicker: Thành phần giao diện để người dùng chọn giờ và phút cho báo thức.
* btnSetAlarm: Nút để người dùng nhấn và cài đặt báo thức với thời gian đã chọn trên timePicker
  + **Phương thức:**

**onCreate**: Phương thức được gọi khi AlarmActivity được tạo ra. Thiết lập giao diện người dùng), gán TimePicker và Button từ giao diện. Đặt chế độ hiển thị 24 giờ cho timePicker và gán sự kiện nhấn cho btnSetAlarm, để khi nhấn sẽ thực hiện cài đặt báo thức.

**setAlarm()**: thiết lập báo thức.

* **Lớp CreateTaskActivity**
  + **Thuộc tính:**
* etTaskName: Trường nhập tên công việc.
* etTaskDescription: Trường nhập mô tả chi tiết công việc.
* etDate: Trường nhập ngày thực hiện công việc, mở ra DatePicker khi được nhấp.
* etStartTime: Trường nhập thời gian bắt đầu công việc, mở ra TimePicker khi được nhấp.
* etEndTime: Trường nhập thời gian kết thúc công việc, mở ra TimePicker khi được nhấp.etLocation: Trường nhập vị trí công việc.
* spAlarmTime: Menu chọn thời gian thông báo cho công việc (trước bao lâu).spColor: Menu chọn màu sắc biểu tượng cho công việc.
* imageColor: Biểu tượng màu hiển thị cho công việc được chọn từ spColor.
* btnCreateTask: Nút tạo mới công việc.
* img\_buttonback: Nút trở về màn hình trước đó.
* id: ID duy nhất của công việc.
* idCurrentUser: ID của người dùng hiện tại.
  + **Phương thức:**
* onCreate: Phương thức khởi tạo Activity. Thiết lập giao diện người dùng, khởi tạo các trường dữ liệu, hiển thị DatePicker và TimePicker khi người dùng nhấp vào các trường tương ứng, và đặt sự kiện cho nút tạo công việc.
* initUi(): Khởi tạo và ánh xạ các thành phần giao diện từ file layout.
* addTask(): Phương thức thực hiện thêm công việc mới. Kiểm tra tính hợp lệ của dữ liệu nhập, sau đó thêm công việc vào cơ sở dữ liệu nếu công việc không trùng lặp. Sau khi thêm thành công, công việc được cài đặt thông báo và chuyển hướng về màn hình chính.
* getNextId(): Lấy ID tiếp theo cho công việc bằng cách tìm ID lớn nhất hiện tại từ cơ sở dữ liệu.
* showDatePickerDialog: Hiển thị DatePicker cho người dùng chọn ngày và thiết lập kết quả vào EditText được truyền vào.
* showTimePickerDialog: Hiển thị TimePicker cho người dùng chọn giờ và thiết lập kết quả vào EditText được truyền vào.
* showAlarmTime(): Hiển thị các tùy chọn thời gian thông báo trong Spinner cho người dùng chọn.
* showColor(): Hiển thị các tùy chọn màu sắc trong Spinner, và cập nhật biểu tượng màu khi người dùng chọn màu.
* setupBackButton(): Thiết lập sự kiện cho nút trở về, giúp người dùng quay lại màn hình trước đó.
* **Lớp UpdateTaskActivity**
  + **Thuộc tính:**
* etTaskName (EditText): Trường nhập tên công việc.
* etTaskDescription (EditText): Trường nhập mô tả công việc.
* etDate (EditText): Trường nhập ngày của công việc.
* etStartTime (EditText): Trường nhập giờ bắt đầu.
* etEndTime (EditText): Trường nhập giờ kết thúc.
* etLocation (EditText): Trường nhập địa điểm công việc.
* spAlarmTime (Spinner): Menu chọn thời gian nhắc nhở.
* spColor (Spinner): Menu chọn màu sắc của công việc.
* imageColor (ImageView): Hình ảnh màu sắc liên quan đến loại công việc.
* id (int): Mã định danh của công việc.
* idCurrentUser (String): Mã định danh người dùng hiện tại.
* btnUpdateTask (Button): Nút để cập nhật công việc.
* img\_buttonback (ImageButton): Nút để quay lại màn hình trước đó.
  + **Phương thức:**
* **onCreate(Bundle savedInstanceState):** Phương thức được gọi khi Activity khởi tạo. Thiết lập giao diện và các sự kiện liên quan.
* **initUi()**: Khởi tạo các thuộc tính giao diện người dùng và thiết lập bộ lọc cho các trường nhập liệu.
* **getTask()**: Nhận dữ liệu công việc từ Intent và hiển thị thông tin lên giao diện người dùng để chỉnh sửa.
* **editTask():** Chỉnh sửa thông tin công việc. Gồm kiểm tra dữ liệu đầu vào, cập nhật công việc vào cơ sở dữ liệu, thiết lập lại thông báo nếu có, và điều hướng về màn hình chính.
* **showAlarmTime()**: Hiển thị các tùy chọn thời gian nhắc nhở bằng Spinner.
* **showColor()**: Hiển thị các tùy chọn màu sắc bằng Spinner và thiết lập màu sắc tương ứng trong ImageView.
* **setupBackButton()**: Thiết lập hành động khi nút quay lại được nhấn, điều hướng về màn hình trước đó.
* **Lớp NotificationHelper**
  + **Phương thức:**
* setAlarm: Thiết lập một báo thức (Alarm) cho công việc Task dựa trên thời gian nhắc nhở của công việc đó. Phương thức sử dụng AlarmManager để tạo báo thức và phát Intent đến NotificationReceiver.
* cancelAlarm: Hủy bỏ báo thức đã được thiết lập trước đó dựa trên taskId.
* **Lớp NotificationReceiver**
  + **Phương thức**: onReceive: xử lý thông báo dựa trên thông tin từ công việc (Task) và hiển thị một thông báo (Notification) cho người dùng.
* **Lớp MapActivity**
  + **Thuộc tính:**
* mapView: Quản lý và hiển thị bản đồ.
* gestureDetector: Xử lý các cử chỉ trên bản đồ, đặc biệt là sự kiện double-tap.

**Phương thức:**

* onCreate(): Khởi tạo giao diện và các cài đặt ban đầu.
* addMarkerAtLocation(): Thêm marker tại vị trí được chọn trên bản đồ.
* onPause(): Dừng hoạt động của bản đồ khi tạm dừng Activity.
* onResume(): Khởi động lại bản đồ khi Activity tiếp tục hoạt động.
* **Lớp TimerActivity**
  + **Thuộc tính:**
  + DEFAULT\_TIME\_IN\_MILLIS: Giá trị thời gian đếm ngược mặc định.
  + tvTimer: Hiển thị thời gian đếm ngược.
  + btnSetTime: Nút để mở hộp thoại chọn thời gian.
  + btnStartReset: Nút để bắt đầu hoặc đặt lại đồng hồ đếm ngược.
  + countDownTimer: Đối tượng xử lý việc đếm ngược.
  + isRunning: Trạng thái của bộ đếm ngược (đang chạy hoặc không).
  + timeLeftInMillis: Thời gian còn lại của bộ đếm ngược
  + **Phương thức:**
* onCreate(): Thiết lập giao diện và sự kiện người dùng.
* openTimePickerDialog(): Mở hộp thoại chọn thời gian.
* startTimer(): Bắt đầu đếm ngược.
* resetTimer(): Đặt lại đồng hồ về giá trị mặc định.
* updateTimerUI(): Cập nhật giao diện bộ đếm ngược.
* onDestroy(): Hủy bỏ đếm ngược khi Activity bị hủy.
* **Lớp MainActivity**
  + **Thuộc tính:**
* CHANNEL\_ID: ID của kênh thông báo.
* bottomNavigationView: Thanh điều hướng cho người dùng.
* frameLayout: Container cho các Fragment.
* toolbar: Thanh công cụ chính của ứng dụng.
  + **Phương thức:**
* onCreate(): Thiết lập giao diện và xử lý intent để tải fragment.
* onCreateOptionsMenu(): Tạo menu tùy chọn cho thanh công cụ.
* onOptionsItemSelected(): Xử lý sự kiện khi mục menu được chọn.
* loadFragment(): Tải Fragment vào frameLayout.
* createNotificationChannel(): Tạo kênh thông báo cho ứng dụng.

**Mối quan hệ giữa các lớp:**

* MainActivity sử dụng CalendarFragment, TaskFragment, WeatherFragment, và các lớp Activity khác để điều hướng giữa các giao diện người dùng và chức năng của ứng dụng.
* NotificationHelper và NotificationReceiver quản lý việc tạo và xử lý thông báo cho các Task và Event.
* TaskDao và UserDao được sử dụng bởi các lớp như CreateTaskActivity, UpdateTaskActivity, và các màn hình người dùng khác để quản lý dữ liệu nhiệm vụ và thông tin người dùng.
* TimerActivity tương tác với người dùng để thiết lập và quản lý đồng hồ đếm ngược.
* MapActivity sử dụng GeoPoint và Marker từ thư viện bản đồ để hiển thị vị trí trên bản đồ cho người dùng.
* EventAdapter và TaskAdapter quản lý việc hiển thị danh sách Event và Task trong các Fragment tương ứng.Lưu ý: Biểu đồ lớp có thể được vẽ chi tiết hơn bằng cách sử dụng UML (Unified Modeling Language) để thể hiện rõ ràng các thuộc tính, phương thức, và mối quan hệ giữa các lớp.

**Thiết kế cơ sở dữ liệu:**

https://console.firebase.google.com/u/0/project/calendar-82deb/overview

**Thiết kế giao diện:**

* Đăng ký
* Đăng nhập
* Đăng xuất
* Quản lý trang cá nhân:
* Thay đổi avatar
* Quên mật khẩu
* Báo thức
* Thời tiết
* Xem ngày âm lịch
* Xem ngày lễ
* Bấm giờ
* Danh sách việc cần làm

**2.3. Triển khai:**

* Viết code: Sử dụng Java để cài đặt các class trong mô hình MVC, đọc/ghi file, xử lý dữ liệu và hiển thị giao diện console.
* Chúng ta sẽ triển khai các lớp theo mô hình MVC (Model-View-Controller) để quản lý dữ liệu và giao diện console của ứng dụng.
* Model: Lớp Task sẽ đại diện cho các nhiệm vụ.
* View: Lớp ConsoleView để hiển thị thông tin cho người dùng.
* Controller: Lớp TaskController sẽ xử lý dữ liệu và tương tác với các nhiệm vụ.

**Kiểm thử:**

Test Case: Đăng ký thành công với thông tin hợp lệ

* ID: 01
* Mục tiêu: Kiểm tra đăng ký thành công với thông tin hợp lệ.
* Điều kiện tiền đề: Màn hình đăng ký đã được mở.
* Các bước thực hiện:
  1. Nhập tên người dùng hợp lệ (ít nhất 6 ký tự).
  2. Nhập địa chỉ email hợp lệ (ví dụ: "test@example.com").
  3. Nhập mật khẩu với độ dài tối thiểu 6 ký tự.
  4. Nhấn nút "Đăng ký".
* Kết quả mong đợi:
  1. Hiện thông báo "User created".
  2. Hiện thông báo xác nhận email "Verification Email Has been Sent".
  3. Chuyển hướng đến màn hình chính của ứng dụng.

**Test Case: Đăng nhập với thông tin hợp lệ**

* ID: 02
* Mục tiêu: Kiểm tra xem người dùng có thể đăng nhập với email và mật khẩu hợp lệ.
* Điều kiện tiền đề:
  + Người dùng đã đăng ký và có email và mật khẩu hợp lệ.
  + Firebase được cấu hình đúng.
* Các bước thực hiện:
  + Mở màn hình đăng nhập.
  + Nhập email và mật khẩu hợp lệ.
  + Nhấn nút "Đăng nhập".
* Kết quả mong đợi:
  + Người dùng sẽ đăng nhập thành công và chuyển hướng tới MainActivity.

**Test Case: Thêm công việc mới với thông tin hợp lệ**

* ID: 03
* Mục tiêu: Kiểm tra xem người dùng có thể thêm công việc mới với thông tin hợp lệ.
* Điều kiện tiền đề:
  + Người dùng đã đăng nhập
  + Firebase và cơ sở dữ liệu đã được cấu hình đúng.
* Các bước thực hiện:

1. Mở màn hình thêm công việc

2. Nhập tên công việc

3. Nhập mô tả công việc

4. Chọn ngày hợp lệ cho công việc

5. Nhập thời gian bắt đầu và kết thúc hợp lệ

6. Chọn thời gian thông báo

7. Chọn màu sắc phù hợp

8. Nhập vị trí công việc

9. Nhấn nút "Thêm công việc".

* Kết quả mong đợi:
  + Công việc mới sẽ được thêm thành công vào cơ sở dữ liệu.
  + Thông báo thành công hiển thị: "Công việc đã được thêm thành công".
  + Màn hình chuyển hướng về MainActivity với danh sách các công việc đã cập nhật.

**Test Case: Xem thời tiết cho thành phố đã chọn**

* ID: 04
* Mục tiêu: Kiểm tra xem ứng dụng có thể hiển thị dữ liệu thời tiết chính xác cho thành phố đã chọn từ danh sách.
* Điều kiện tiền đề:
  + Người dùng đã đăng nhập vào hệ thống
  + Ứng dụng đã được kết nối Internet.
  + API key hợp lệ và dịch vụ thời tiết (WeatherService) hoạt động.
* Các bước thực hiện:

1. Mở màn hình thời tiết

2. Chọn thành phố "Hà Nội" từ Spinner

* Kết quả mong đợi:
  + Ứng dụng hiển thị thông tin thời tiết đúng cho thành phố được chọn, bao gồm:
    - Tên thành phố ("Thành phố: Hà Nội").
    - Nhiệt độ (ví dụ: "Nhiệt độ: 25°C").
    - Mô tả thời tiết (ví dụ: "Mô tả: Mưa nhẹ").
    - Icon thời tiết

**Test Case: Thêm báo thức**

* ID: 05
* Mục tiêu: Kiểm tra xem người dùng có thể cài đặt và hủy báo thức thành công.
* Điều kiện tiền đề:
  + Người dùng đã đăng nhập vào hệ thống
  + Thiết bị đã được cấp quyền báo thức và cho phép ứng dụng hoạt động trong nền.
  + AlarmReceiver đã được cấu hình để nhận thông báo báo thức.
* Các bước thực hiện:

1. Mở màn hình báo thức

2. Chọn giờ và phút trong TimePicker

3. Nhấn nút "Cài báo thức"

4. Quan sát thông báo xác nhận và thời gian báo thức đã được đặt.

5. Nhấn nút "Hủy báo thức"

* Kết quả mong đợi:
  + Sau khi nhấn "Cài báo thức," người dùng nhận được thông báo thành công, và báo thức sẽ được phát khi đến thời gian đã đặt.
  + Âm thanh báo thức được phát khi đến thời gian đặt báo thức.
  + Sau khi nhấn "Hủy báo thức," báo thức sẽ bị hủy và âm thanh dừng lại (nếu đang phát).

**Test Case: Xem bản đồ và thêm vị trí**

* ID: 06
* Mục tiêu: Kiểm tra xem người dùng có thể xem bản đồ và thêm marker
* Điều kiện tiền đề:
  + Người dùng đã đăng nhập vào hệ thống
  + Thiết bị có kết nối Internet.
* Các bước thực hiện:

1. Mở màn hình map

2. Quan sát xem bản đồ được hiển thị với vị trí mặc định tại Hà Nội.

3. Chạm hai lần vào một vị trí bất kỳ trên bản đồ.

4. Quan sát bản đồ để xác nhận marker đã được thêm tại vị trí vừa double-tap.

* Kết quả mong đợi:
  + Bản đồ được tải thành công, với vị trí mặc định tại Hà Nội.
  + Sau khi người dùng thực hiện thao tác double-tap, một marker mới sẽ xuất hiện tại vị trí đó trên bản đồ, với tiêu đề hiển thị là “Điểm đã chọn.”
  + Nếu người dùng double-tap ở vị trí khác, marker cũ sẽ được xóa, và một marker mới sẽ xuất hiện tại vị trí vừa chọn.

**Test case: Bấm giờ**

* ID: 07
* Mục tiêu: Bấm giờ đếm ngược
* Điều kiện tiền đề:
  + Người dùng đã đăng nhập vào hệ thống
* Các bước thực hiện :

1. Mở màn hình bấm giờ

2. Ấn nút “Đặt thời gian”

3. Chọn giờ, phút và giây từ NumberPicker và nhấn "Xác nhận".

4. Chọn chuông báo thức

5. Ấn nút play

* Kết quả mong đợi:
* Bộ đếm thời gian bắt đầu đếm ngược, cập nhật thời gian trên giao diện mỗi giây.

Test case: Đặt lại mật khẩu

* ID: 08
* Mục tiêu: Đặt lại mật khẩu của tài khoản người dùng
* Điều kiện tiền đề:
  + Người dùng đã đăng nhập vào hệ thống
* Các bước thực hiện :

1. Mở màn hình người dùng

2. Ấn nút “Đặt lại mật khẩu”

3. Nhập mật khẩu mới

4. Ấn nút “OK”

* Kết quả mong đợi:
* Hệ thống hiện thông báo Password Successfully Updated

Test case: Đăng xuất

* ID: 09
* Mục tiêu: Thoát khỏi tài khoản đang sử dụng
* Điều kiện tiền đề:
  + Người dùng đã đăng nhập vào hệ thống
* Các bước thực hiện :

1. Mở màn hình người dùng

2. Ấn nút “Đăng xuất”

* Kết quả mong đợi:
* Hệ thống hiện thông báo Success Logout
* Màn hình hiển thị giao diện đăng nhập

**2.4. Vận hành và bảo trì:**

* Vận hành: Ứng dụng lịch sau khi triển khai cần được vận hành một cách liên tục và ổn định để đáp ứng nhu cầu của người dùng.
* Theo dõi hiệu suất hệ thống: Sử dụng các công cụ giám sát để kiểm tra hoạt động của ứng dụng, như việc tải dữ liệu lịch, nhắc nhở, hoặc thông báo đẩy. Phân tích log hệ thống định kỳ để phát hiện sớm các lỗi tiềm ẩn hoặc các vấn đề về hiệu suất.
* Quản lý tài khoản người dùng: Theo dõi đăng nhập, đăng ký, và xác thực tài khoản người dùng. Quản lý quyền truy cập và đảm bảo tính bảo mật thông qua các phương pháp xác thực an toàn.
* Sao lưu dữ liệu: Đảm bảo dữ liệu lịch, sự kiện của người dùng được sao lưu thường xuyên, đặc biệt là dữ liệu quan trọng như các sự kiện, nhiệm vụ đã lưu. Sao lưu định kỳ sẽ giúp giảm thiểu mất mát dữ liệu khi có sự cố xảy ra.
* Cập nhật định kỳ: Đảm bảo ứng dụng được cập nhật với các tính năng mới và sửa lỗi. Cập nhật các API bên thứ ba như Google Calendar API nếu có sự thay đổi về phiên bản hoặc chính sách.
* Bảo trì:
* Bảo trì cơ sở dữ liệu: Kiểm tra tính toàn vẹn của dữ liệu, loại bỏ các sự kiện hoặc dữ liệu lỗi thời, tối ưu hóa truy vấn và hiệu suất của cơ sở dữ liệu để đảm bảo thời gian phản hồi nhanh.
* Sửa lỗi (bug fixing): Khi phát hiện các lỗi phát sinh trong quá trình sử dụng, cần nhanh chóng phân tích và sửa chữa. Việc sửa lỗi nên được thực hiện dựa trên mức độ nghiêm trọng và ảnh hưởng của lỗi đến người dùng.
* Nâng cấp bảo mật: Liên tục theo dõi và nâng cấp các thành phần bảo mật của ứng dụng, từ mã hóa dữ liệu người dùng cho đến bảo mật kết nối API và hệ thống xác thực. Đảm bảo rằng ứng dụng tuân thủ các tiêu chuẩn bảo mật hiện hành.
* Cải tiến giao diện và trải nghiệm người dùng: Lắng nghe phản hồi từ người dùng để liên tục cải thiện giao diện và tính năng của ứng dụng, đảm bảo rằng trải nghiệm người dùng luôn được nâng cao.

**CHƯƠNG 3. KẾT QUẢ THỰC HIỆN**

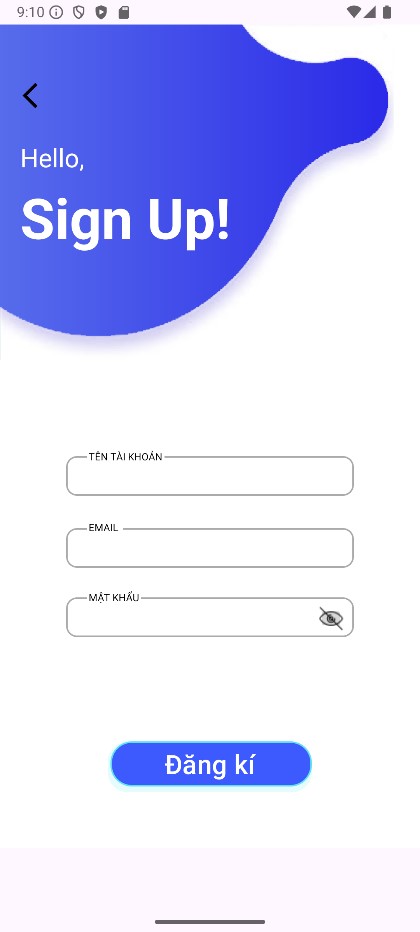
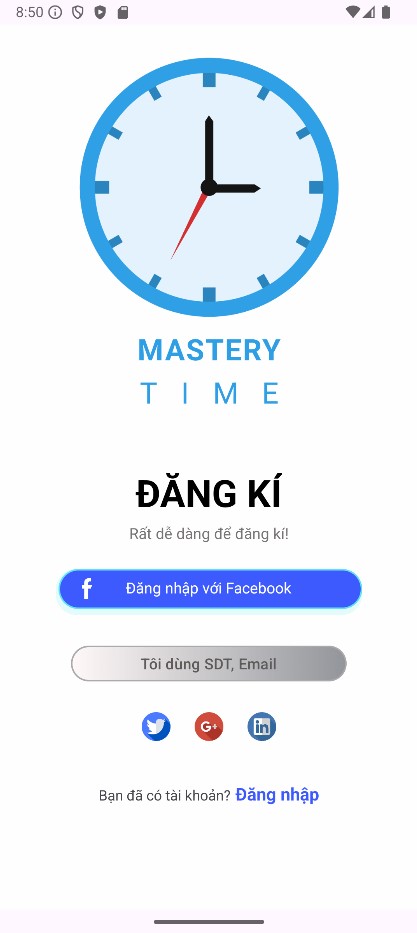
**3.1. Công nghệ đã sử dụng**

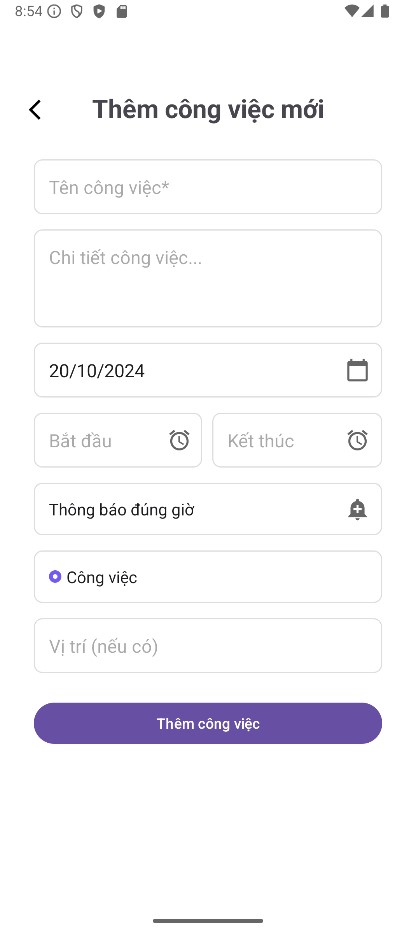
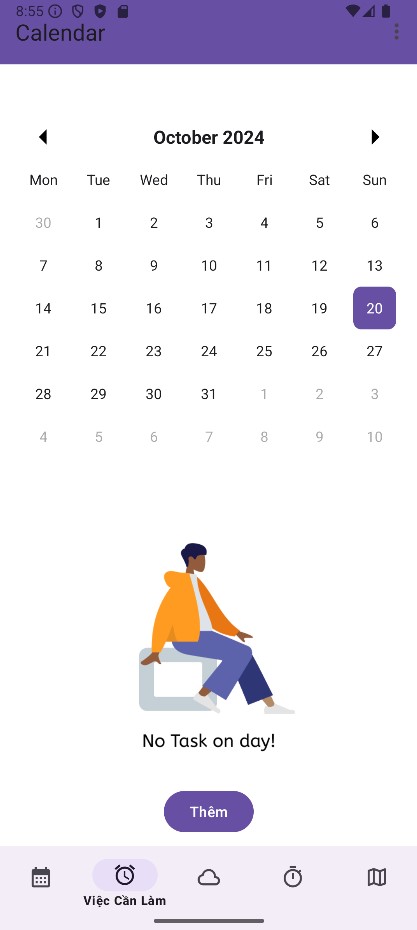
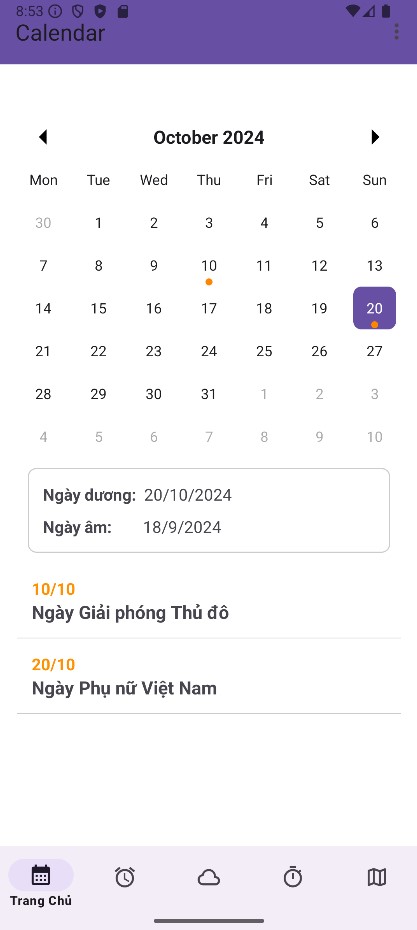
* Ngôn ngữ lập trình: Java
* Công cụ: Android studio

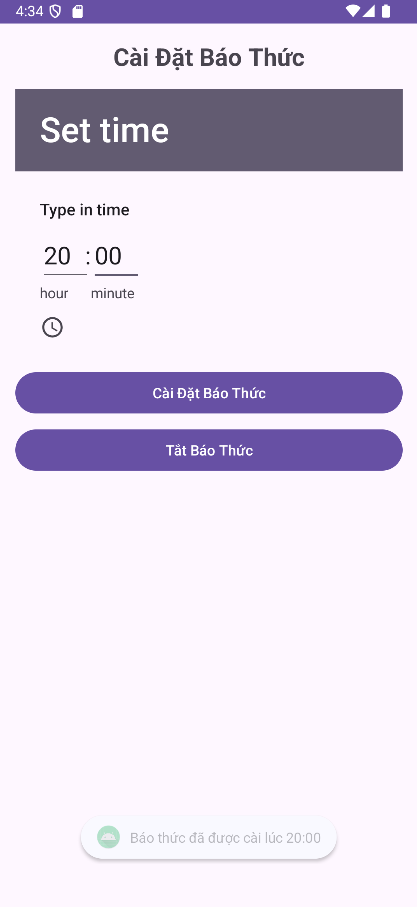
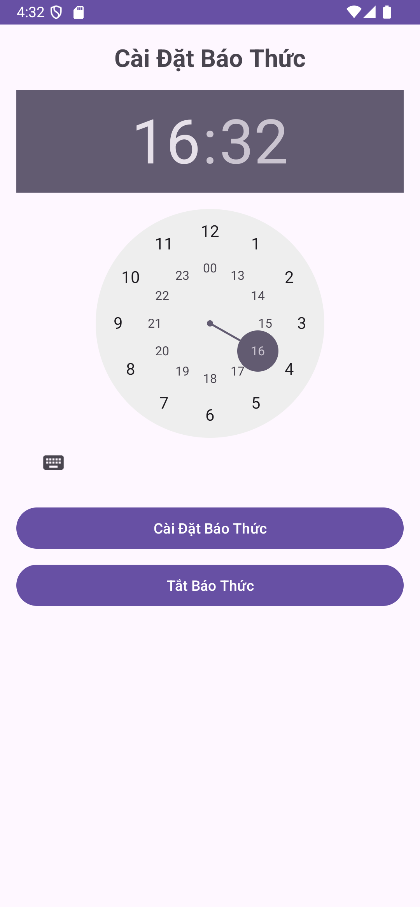
**3.2. Tiến độ thực hiện**

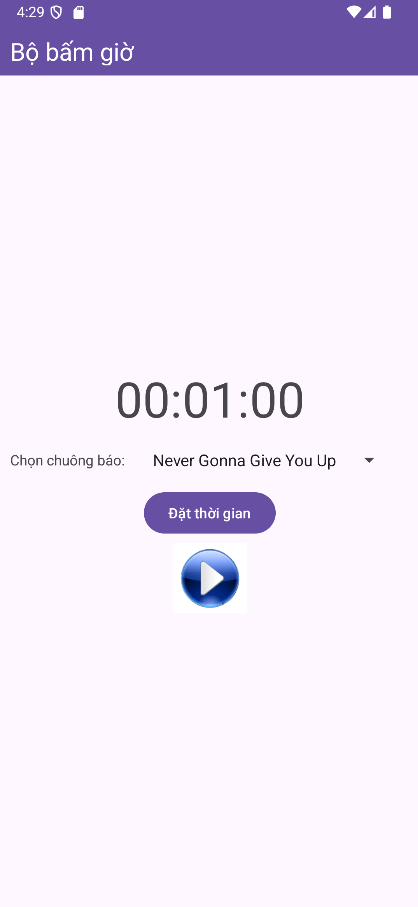
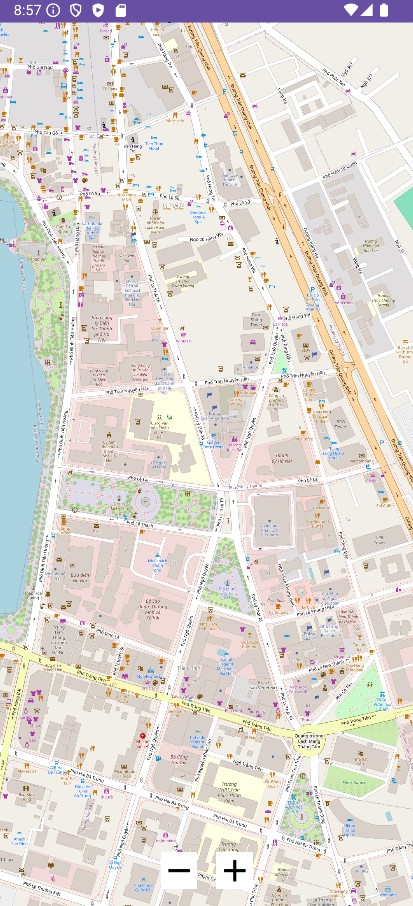
Link github tới dự án: https://github.com/NguyenDuyChinhK/CSE441\_PROJECT.git

**3.3. Hình ảnh sản phẩm**

****

****

****

****

**KẾT LUẬN**

Ứng dụng lịch mang lại một số điểm mạnh khi sử dụng ứng dụng lịch trong quản lý thời gian cá nhân và công việc: người dùng có thể tạo và quản lý các sự kiện, nhắc nhở; ứng dụng tích hợp các tính năng thông báo, đồng bộ hóa với lịch Google; giao diện thân thiện giúp sắp xếp, theo dõi công việc hàng ngày một cách dễ dàng. Do đó, thông báo có thể được tùy biến và nhiều nhiệm vụ quản lý, giúp người dùng có thể làm việc hiệu quả và tổ chức cuộc sống của họ.

Tuy nhiên, vẫn tồn tại một số nhược điểm trong quá trình phát triển và sử dụng ứng dụng lịch. Một số vấn đề có thể phát sinh từ hiệu suất hoạt động, như ứng dụng chạy chậm hoặc tiêu tốn tài nguyên của thiết bị khi quản lý lượng lớn sự kiện. Vấn đề bảo mật cũng cần được chú trọng, đặc biệt là khi ứng dụng yêu cầu quyền truy cập vào thông tin cá nhân hoặc liên kết với các dịch vụ bên thứ ba như Google Calendar. Khả năng mở rộng và nâng cấp ứng dụng để đáp ứng các nhu cầu đa dạng của người dùng vẫn còn là một thách thức đối với đội ngũ phát triển.

Trong tương lai, hướng phát triển của ứng dụng lịch sẽ tập trung vào cải thiện hiệu suất và bảo mật, cũng như nâng cấp các tính năng thông minh như tích hợp trí tuệ nhân tạo để gợi ý lịch trình tối ưu cho người dùng. Việc mở rộng khả năng đồng bộ với các nền tảng khác, cải thiện giao diện người dùng và trải nghiệm tổng thể cũng sẽ là những mục tiêu quan trọng. Ngoài ra, hỗ trợ tốt hơn cho các nhóm người dùng doanh nghiệp và cộng tác trong công việc sẽ giúp ứng dụng tiếp cận được với nhiều phân khúc người dùng hơn.

**DANH MỤC TÀI LIỆU THAM KHẢO**

1. Android Programming: The Big Nerd Ranch Guide - Bill Phillips, Chris Stewart
2. Head First Android Development: A Brain-Friendly Guide - Dawn Griffiths, David Griffiths
3. Google Developers Documentation for Android**:** [developer.android.com](https://developer.android.com/)

[4]. Firebase Documentation: firebase.google.com/docs