Auraria Mobile Parking (AMP)

Metropolitan State University of Denver

William Maddock, Aaron Nkouka, Seth Jones, Milagros Hernandez-Vasquez, Sayizana Worku, Michael Bate

Roles & Responsibilities

Michael Bate – Project Manager, responsible for coordinating the team and ensuring project deadlines are met. Helped test and debug features.

William Maddock – Lead Developer, overseeing technical development and implementation.

Aaron Nkouka – Feature-Driven Development (FDD), focused on key feature implementations.

Seth Jones – Feature-Driven Development (FDD), assisting in developing specific app features.

Milagros Hernandez-Vasquez – Feature-Driven Development (FDD), working on app enhancements.

Sayizana Worku – Feature-Driven Development (FDD), supporting feature creation and refinement.

.

Vision Statement

- Transform the parking experience for students, staff, faculty, and visitors at MSU Denver through a comprehensive mobile solution. Originally we planned

- **Auraria Mobile Parking (AMP)**
  - will empower users to seamlessly navigate and manage parking on the Auraria Campus with ease.

Target Audience

- Built for developers to add investment and improvements
- Once the developers make it more finalized version they can deploy it to Auraria Campus

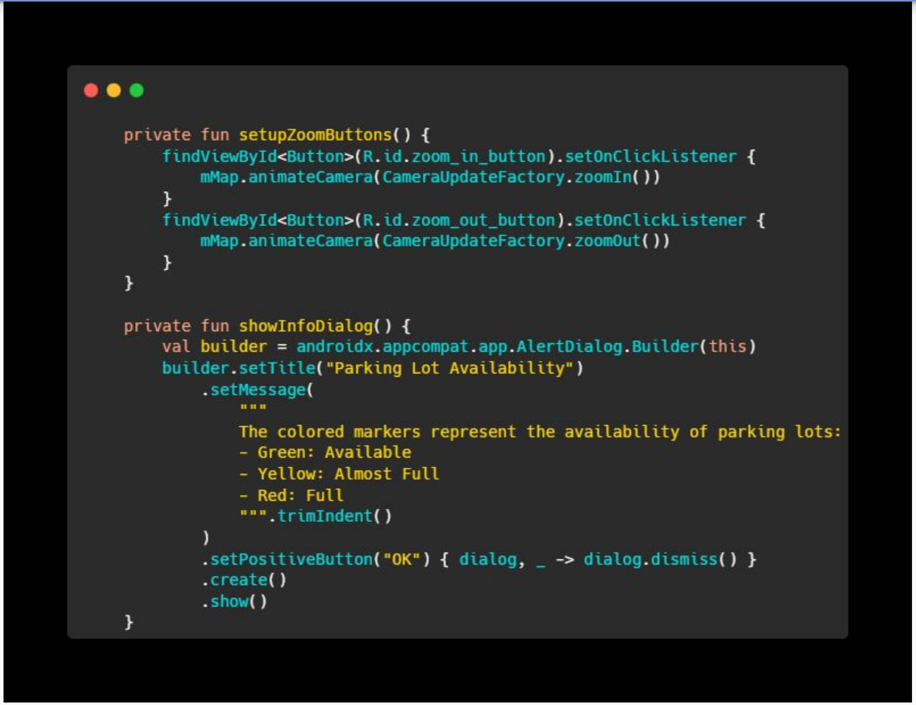
Overview

- AMP offers unique features not found in other parking apps, such as real-time parking availability and lot filtering to find the best options for users. The app provides personalized recommendations through user reviews and includes a parking budget simulator to estimate semester parking costs. These features streamline parking and enhance convenience for students, faculty, and visitors.

Key Features

1. **Real-time Parking Availability**
2. **Filter Parking Lots**
3. **User Reviews and Ratings**
4. **AI Chat bot**
5. **Parking Budget Simulator**
6. **Multi-language Access**
7. **Live Weather Updates**
8. **Live Ball Arena updates**

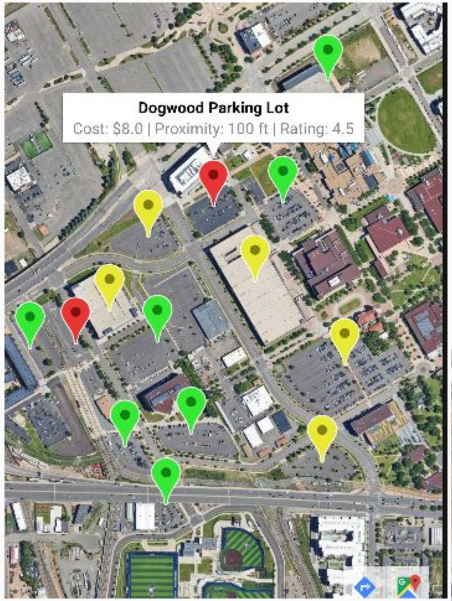
Real-time Parking Availability



```
private fun setupZoomButtons() {
    findViewById<Button>(R.id.zoom_in_button).setOnClickListener {
        mMap.animateCamera(CameraUpdateFactory.zoomIn())
    }
    findViewById<Button>(R.id.zoom_out_button).setOnClickListener {
        mMap.animateCamera(CameraUpdateFactory.zoomOut())
    }
}

private fun showInfoDialog() {
    val builder = androidx.appcompat.app.AlertDialog.Builder(this)
    builder.setTitle("Parking Lot Availability")
        .setMessage(
            """
            The colored markers represent the availability of parking lots:
            - Green: Available
            - Yellow: Almost Full
            - Red: Full
            """.trimIndent()
        )
        .setPositiveButton("OK") { dialog, _ -> dialog.dismiss() }
        .create()
        .show()
}
```

Real-time Parking Availability

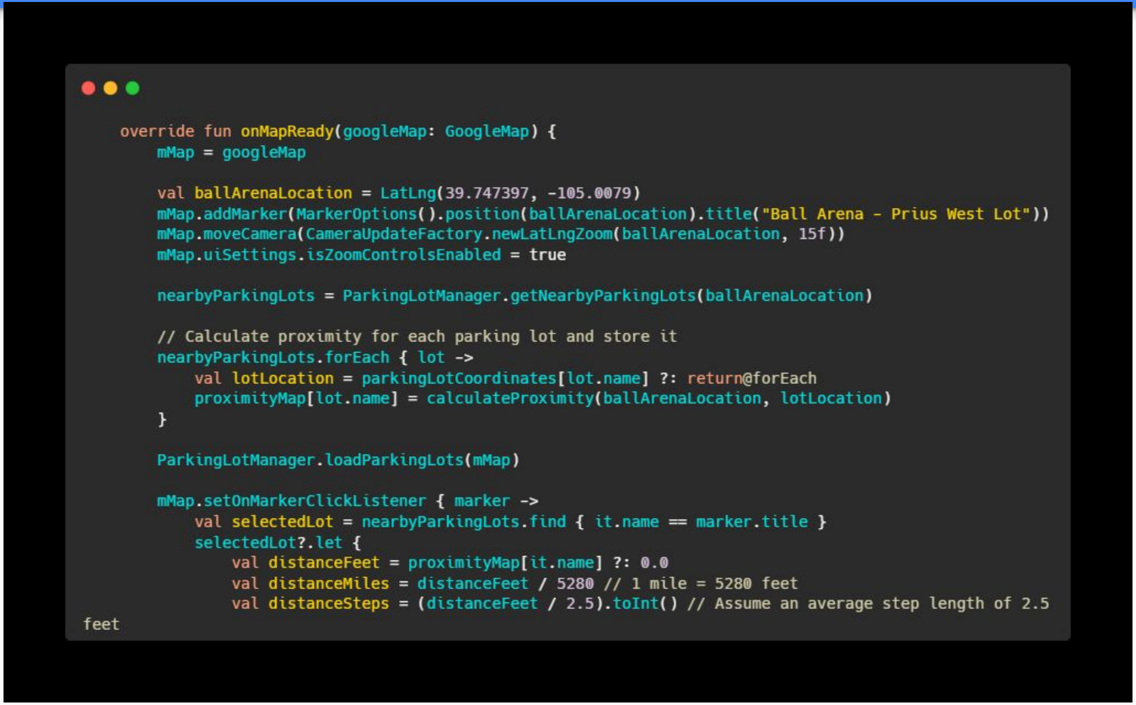
- Save time by checking for open lots before arriving on campus, reducing stress during busy hours.
- Quickly find spots during events or campus visits, improving their overall experience.



Fir Parking Lot
Availability: Available (10%)

Aspen Lot
Availability: Full (85%)

Filter Parking Lots

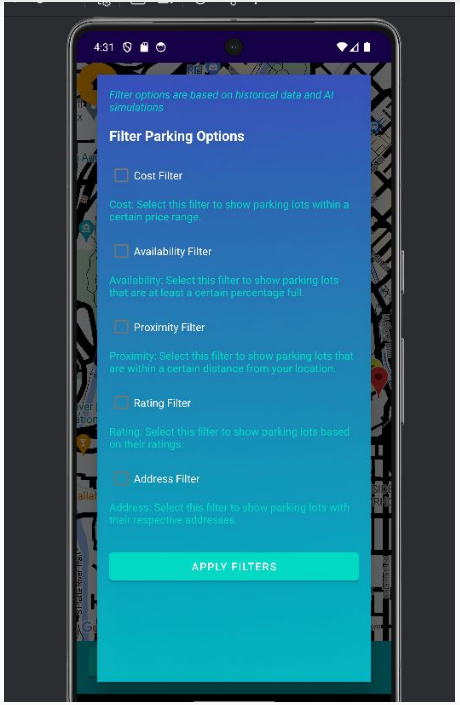


```
override fun onMapReady(googleMap: GoogleMap) {
    mMap = googleMap

    val ballArenaLocation = LatLng(39.747397, -105.0079)
    mMap.addMarker(MarkerOptions().position(ballArenaLocation).title("Ball Arena - Prius West Lot"))
    mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(ballArenaLocation, 15f))
    mMap.uiSettings.isZoomControlsEnabled = true

    nearbyParkingLots = ParkingLotManager.getNearbyParkingLots(ballArenaLocation)

    // Calculate proximity for each parking lot and store it
    nearbyParkingLots.forEach { lot ->
        val lotLocation = parkingLotCoordinates[lot.name] ?: return@forEach
        proximityMap[lot.name] = calculateProximity(ballArenaLocation, lotLocation)
    }

    ParkingLotManager.loadParkingLots(mMap)

    mMap.setOnMarkerClickListener { marker ->
        val selectedLot = nearbyParkingLots.find { it.name == marker.title }
        selectedLot?.let {
            val distanceFeet = proximityMap[it.name] ?: 0.0
            val distanceMiles = distanceFeet / 5280 // 1 mile = 5280 feet
            val distanceSteps = (distanceFeet / 2.5).toInt() // Assume an average step length of 2.5
feet
```

Filter Parking Lots

- Users are able to identify the cheapest and closest lots to their classes, helping to save both money and time.

- Visitors can easily find lots based on proximity to their destination or available spots.



4:31

Filter options are based on historical data and AI simulations

Filter Parking Options

Cost Filter

Cost: Select this filter to show parking lots within a certain price range.

Availability Filter

Availability: Select this filter to show parking lots that are at least a certain percentage full.

Proximity Filter

Proximity: Select this filter to show parking lots that are within a certain distance from your location.

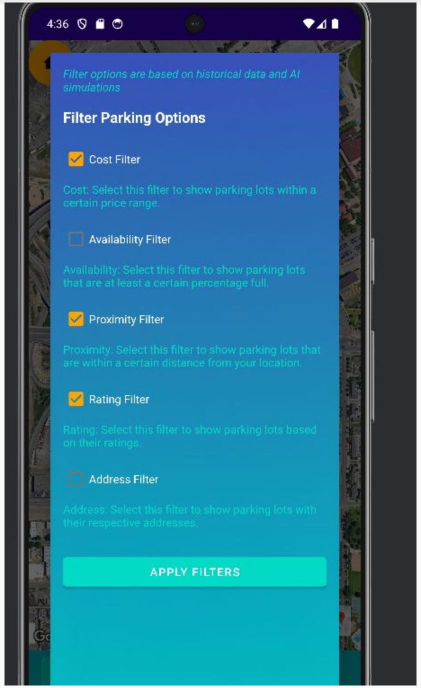
Rating Filter

Rating: Select this filter to show parking lots based on their ratings.

Address Filter

Address: Select this filter to show parking lots with their respective addresses.

APPLY FILTERS



4:36

Filter options are based on historical data and AI simulations

Filter Parking Options

[✓ Cost Filter]

Cost: Select this filter to show parking lots within a certain price range.

Availability Filter

Availability: Select this filter to show parking lots that are at least a certain percentage full.

[✓ Proximity Filter]

Proximity: Select this filter to show parking lots that are within a certain distance from your location.

[✓ Rating Filter]

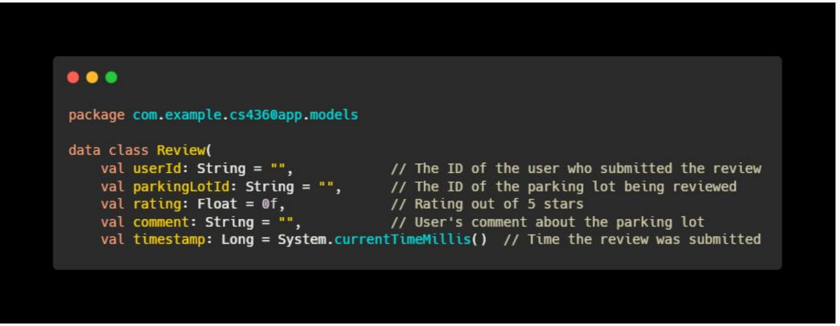
Rating: Select this filter to show parking lots based on their ratings.

Address Filter

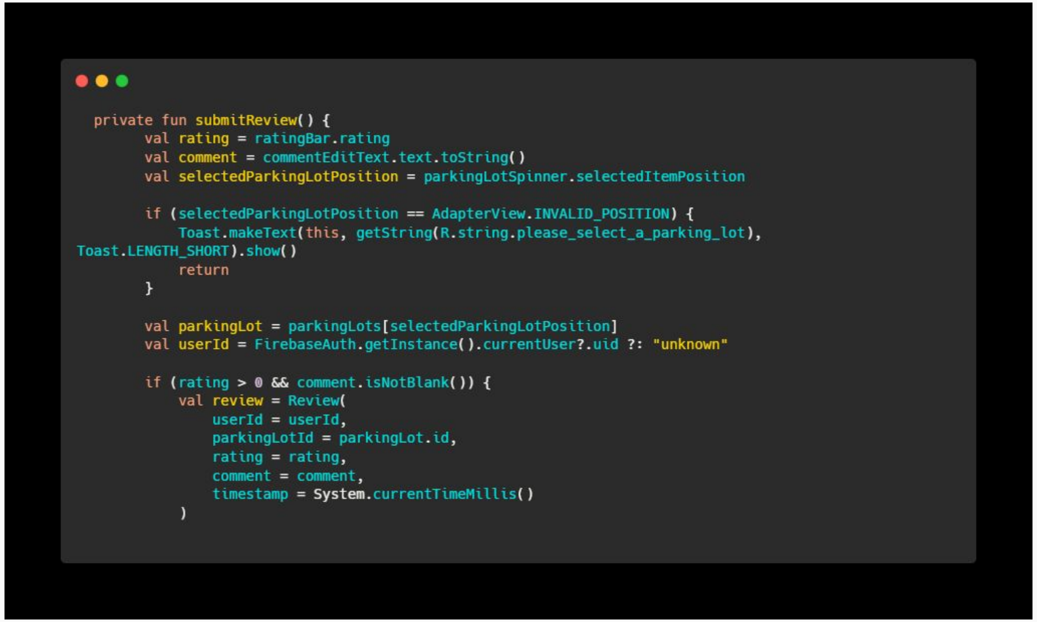
Address: Select this filter to show parking lots with their respective addresses.

APPLY FILTERS

User Reviews and Ratings



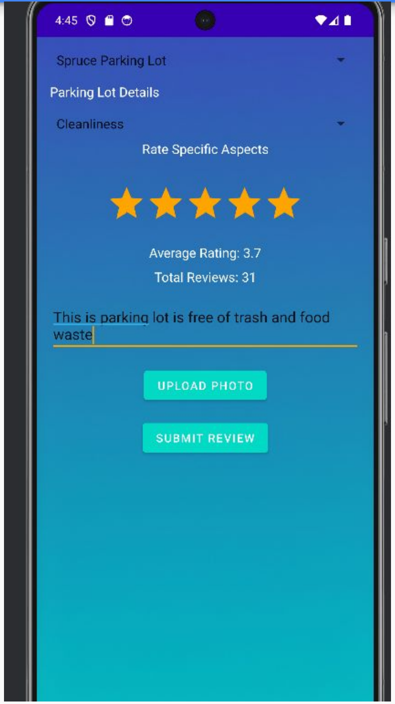
```
package com.example.cs4360app.models

data class Review(
    val userId: String = "",            // The ID of the user who submitted the review
    val parkingLotId: String = "",      // The ID of the parking lot being reviewed
    val rating: Float = 0f,             // Rating out of 5 stars
    val comment: String = "",           // User's comment about the parking lot
    val timestamp: Long = System.currentTimeMillis()  // Time the review was submitted
```



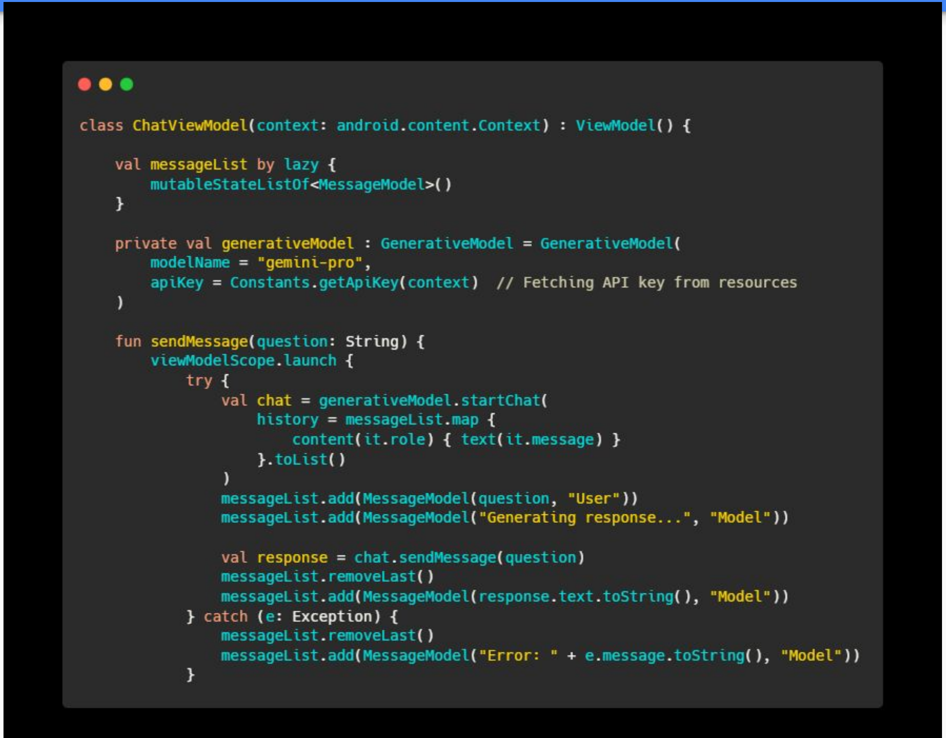
```
  private fun submitReview() {
        val rating = ratingBar.rating
        val comment = commentEditText.text.toString()
        val selectedParkingLotPosition = parkingLotSpinner.selectedItemPosition

        if (selectedParkingLotPosition == AdapterView.INVALID_POSITION) {
            Toast.makeText(this, getString(R.string.please_select_a_parking_lot),
Toast.LENGTH_SHORT).show()
            return
        }

        val parkingLot = parkingLots[selectedParkingLotPosition]
        val userId = FirebaseAuth.getInstance().currentUser?.uid ?: "unknown"

        if (rating > 0 && comment.isNotBlank()) {
            val review = Review(
                userId = userId,
                parkingLotId = parkingLot.id,
                rating = rating,
                comment = comment,
                timestamp = System.currentTimeMillis()
            )
```

User Reviews and Ratings

- Our reviews help our user gather information about other parking lots and garages
- You can use reviews to identify well-maintained or highly rated lots for a positive experience



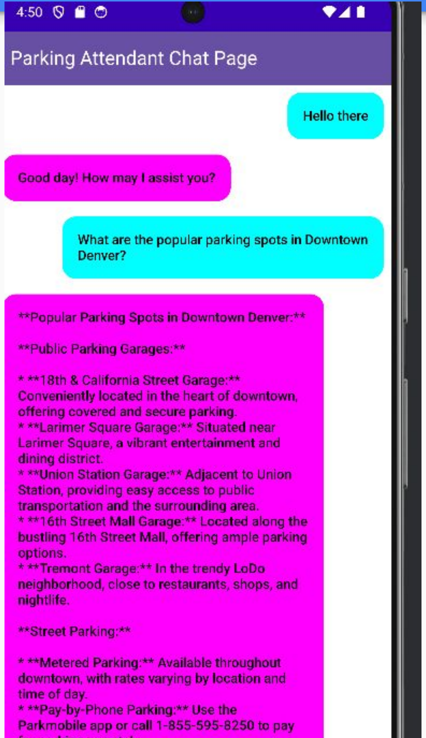
AI chat bot



```kotlin
class ChatViewModel(context: android.content.Context) : ViewModel() {

    val messageList by lazy {
        mutableStateListOf<MessageModel>()
    }

    private val generativeModel : GenerativeModel = GenerativeModel(
        modelName = "gemini-pro",
        apiKey = Constants.getApiKey(context)  // Fetching API key from resources
    )

    fun sendMessage(question: String) {
        viewModelScope.launch {
            try {
                val chat = generativeModel.startChat(
                    history = messageList.map {
                        content(it.role) { text(it.message) }
                    }.toList()
                )
                messageList.add(MessageModel(question, "User"))
                messageList.add(MessageModel("Generating response...", "Model"))

                val response = chat.sendMessage(question)
                messageList.removeLast()
                messageList.add(MessageModel(response.text.toString(), "Model"))
            } catch (e: Exception) {
                messageList.removeLast()
                messageList.add(MessageModel("Error: " + e.message.toString(), "Model"))
            }
```

AI chat bot

- With our AI bot you can quickly ask for directions, updates, or support, avoiding the need to navigate complex menus.
- Receive immediate assistance, making their first-time parking experience seamless.


4:50

Parking Attendant Chat Page

Hello there

Good day! How may I assist you?

What are the popular parking spots in Downtown Denver?

Popular Parking Spots in Downtown Denver:

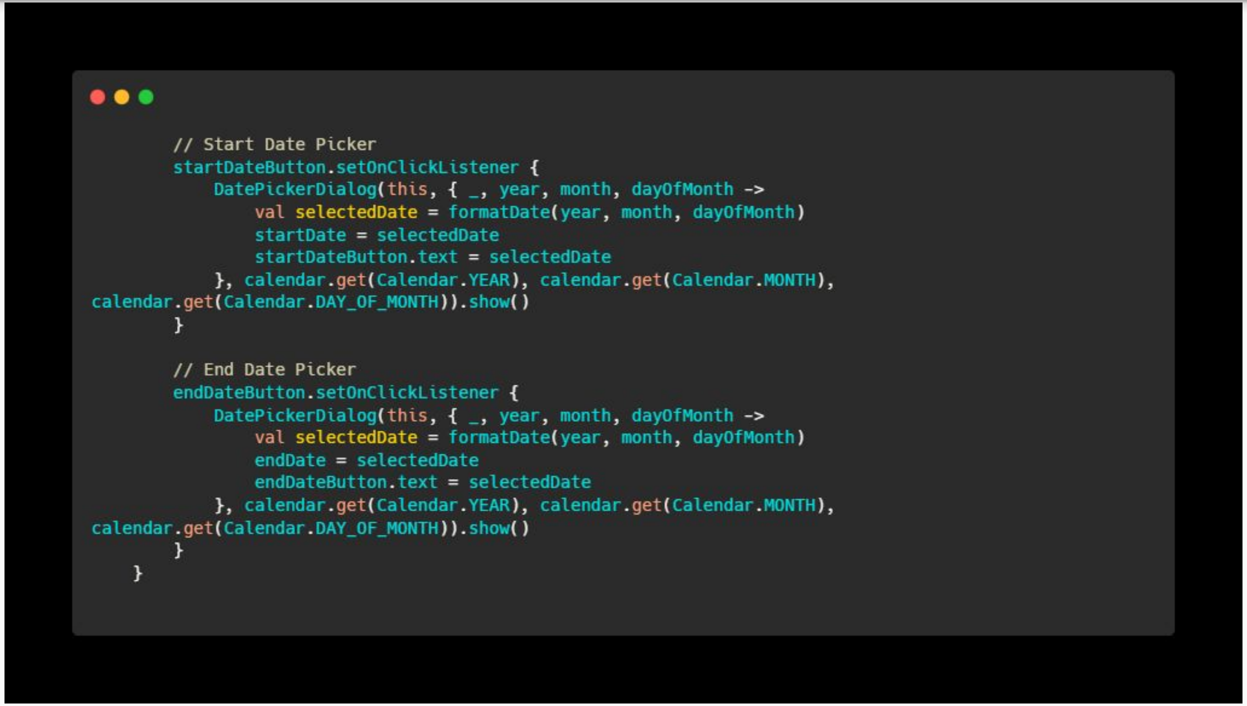
Public Parking Garages:

* **18th & California Street Garage:** Conveniently located in the heart of downtown, offering covered and secure parking.
* **Larimer Square Garage:** Situated near Larimer Square, a vibrant entertainment and dining district.
* **Union Station Garage:** Adjacent to Union Station, providing easy access to public transportation and the surrounding area.
* **16th Street Mall Garage:** Located along the bustling 16th Street Mall, offering ample parking options.
* **Tremont Garage:** In the trendy LoDo neighborhood, close to restaurants, shops, and nightlife.

Street Parking:

* **Metered Parking:** Available throughout downtown, with rates varying by location and time of day.
* **Pay-by-Phone Parking:** Use the Parkmobile app or call 1-855-595-8250 to pay

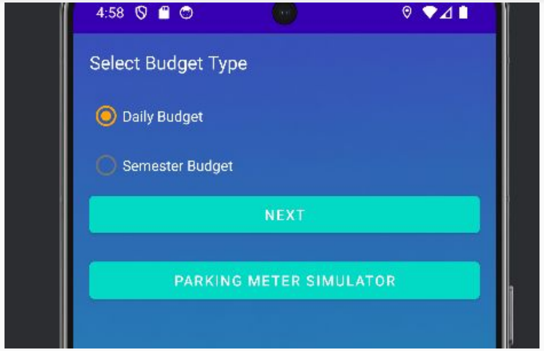
Parking Budget Simulator

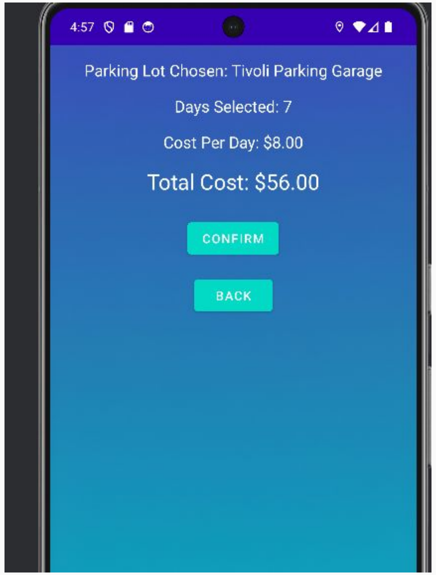


```
        // Start Date Picker
        startDateButton.setOnClickListener {
            DatePickerDialog(this, { _, year, month, dayOfMonth ->
                val selectedDate = formatDate(year, month, dayOfMonth)
                startDate = selectedDate
                startDateButton.text = selectedDate
            }, calendar.get(Calendar.YEAR), calendar.get(Calendar.MONTH),
calendar.get(Calendar.DAY_OF_MONTH)).show()
        }

        // End Date Picker
        endDateButton.setOnClickListener {
            DatePickerDialog(this, { _, year, month, dayOfMonth ->
                val selectedDate = formatDate(year, month, dayOfMonth)
                endDate = selectedDate
                endDateButton.text = selectedDate
            }, calendar.get(Calendar.YEAR), calendar.get(Calendar.MONTH),
calendar.get(Calendar.DAY_OF_MONTH)).show()
        }
    }
```

Parking Budget Simulator

- For students this can help budget their parking expenses efficiently, especially for those commuting daily.
- For staff it help estimate monthly costs, ensuring parking fits within their allocated budgets.
- For people visiting the campus this helps plan the parking expense for specific visits.



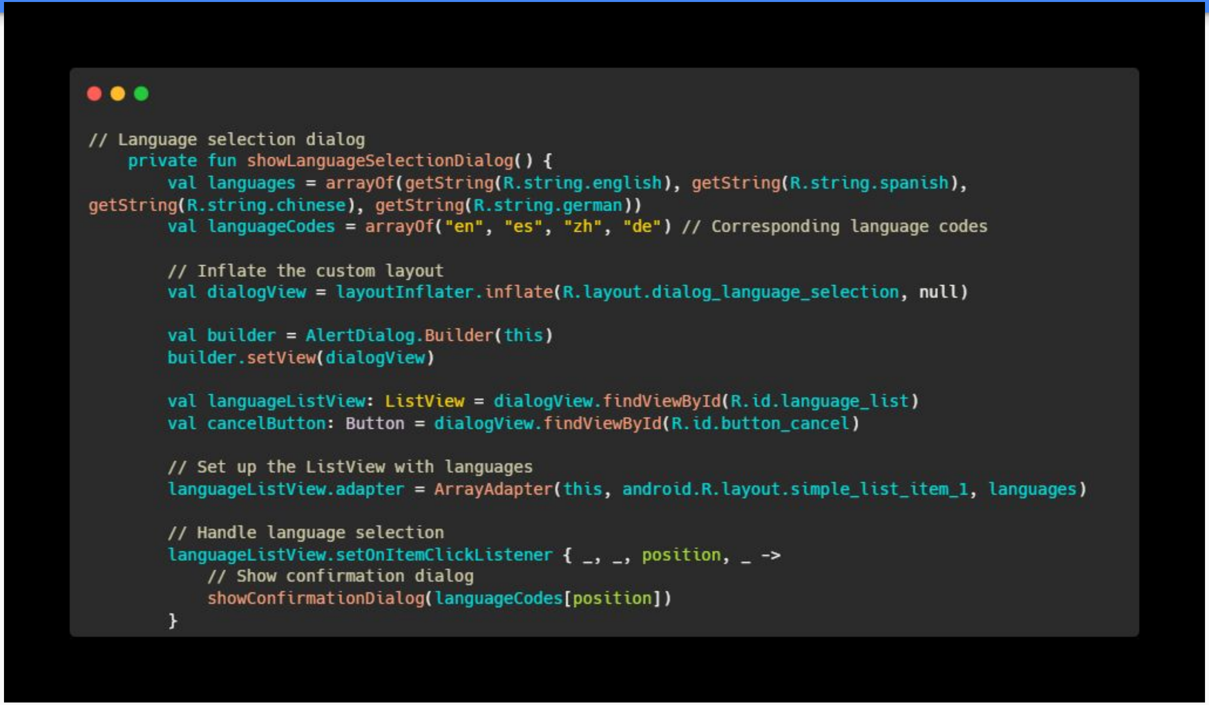

Parking Lot Chosen: Tivoli Parking Garage

Days Selected: 7

Cost Per Day: $8.00

Total Cost: $56.00

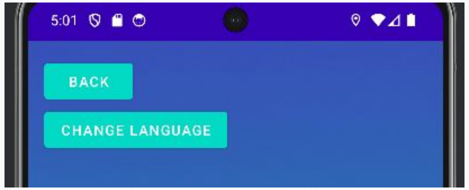
Multi-Language Access

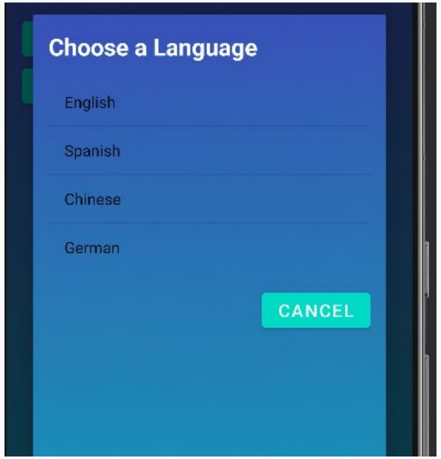


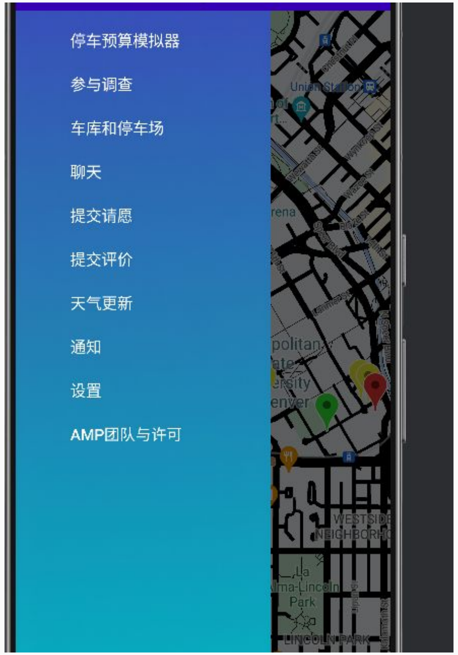
```
// Language selection dialog
    private fun showLanguageSelectionDialog() {
        val languages = arrayOf(getString(R.string.english), getString(R.string.spanish),
getString(R.string.chinese), getString(R.string.german))
        val languageCodes = arrayOf("en", "es", "zh", "de") // Corresponding language codes

        // Inflate the custom layout
        val dialogView = layoutInflater.inflate(R.layout.dialog_language_selection, null)

        val builder = AlertDialog.Builder(this)
        builder.setView(dialogView)

        val languageListView: ListView = dialogView.findViewById(R.id.language_list)
        val cancelButton: Button = dialogView.findViewById(R.id.button_cancel)

        // Set up the ListView with languages
        languageListView.adapter = ArrayAdapter(this, android.R.layout.simple_list_item_1, languages)

        // Handle language selection
        languageListView.setOnItemClickListener { _, _, position, _ ->
            // Show confirmation dialog
            showConfirmationDialog(languageCodes[position])
        }
```

Multi-Language Access

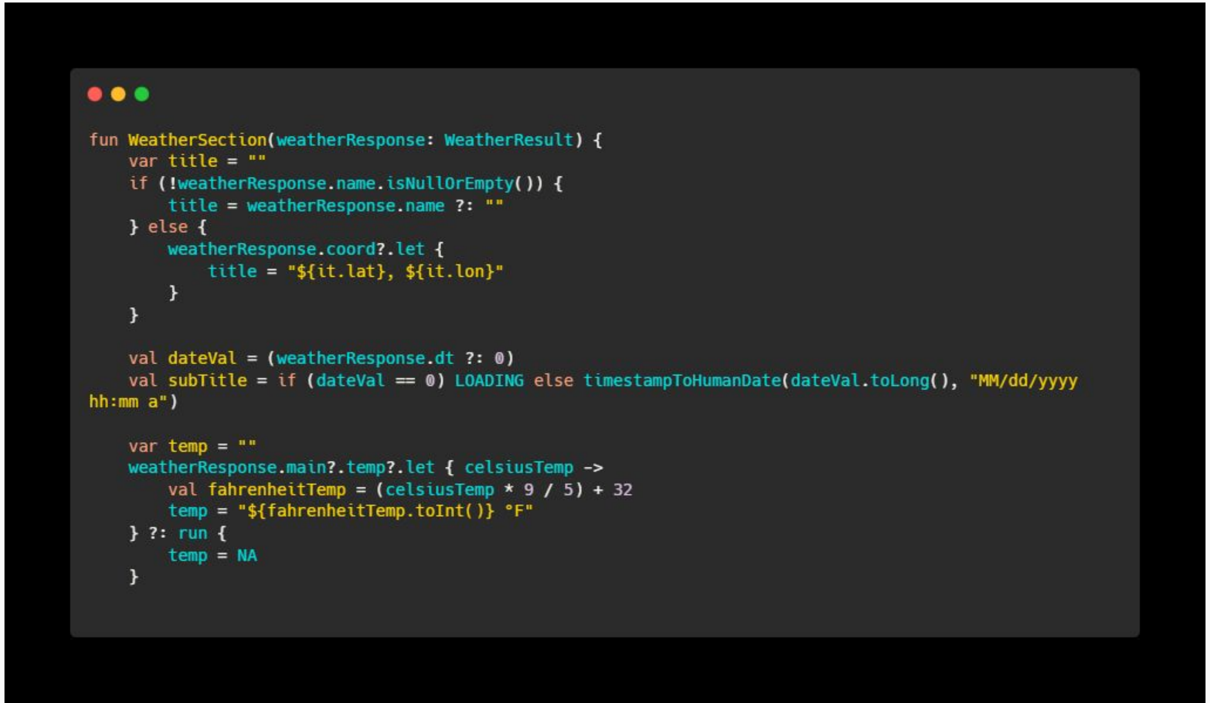
- We have access to multiple languages so that so that users can comfortably navigate the app, fostering inclusivity.
- International faculty members can access the app without language barriers.
- Having our app support multiple languages can help accommodate a diverse audience during conferences or events.







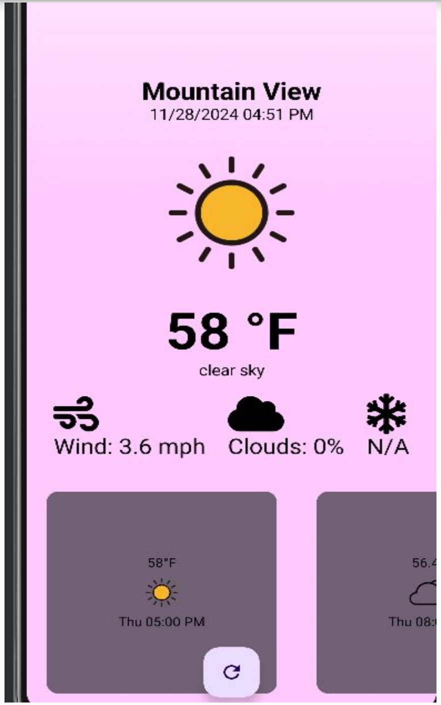
Live Weather Updates



```kotlin
fun WeatherSection(weatherResponse: WeatherResult) {
    var title = ""
    if (!weatherResponse.name.isNullOrEmpty()) {
        title = weatherResponse.name ?: ""
    } else {
        weatherResponse.coord?.let {
            title = "${it.lat}, ${it.lon}"
        }
    }

    val dateVal = (weatherResponse.dt ?: 0)
    val subTitle = if (dateVal == 0) LOADING else timestampToHumanDate(dateVal.toLong(), "MM/dd/yyyy
hh:mm a")

    var temp = ""
    weatherResponse.main?.temp?.let { celsiusTemp ->
        val fahrenheitTemp = (celsiusTemp * 9 / 5) + 32
        temp = "${fahrenheitTemp.toInt()} °F"
    } ?: run {
        temp = NA
    }
```

Live Weather Updates

- Our users can choose parking based on weather, such as avoiding open lots during rain or snow.
- This will help to minimize the exposure to harsh conditions.
- Our users will be able to make informed decisions, ensuring a hassle-free experience regardless of weather.


Mountain View
11/28/2024 04:51 PM

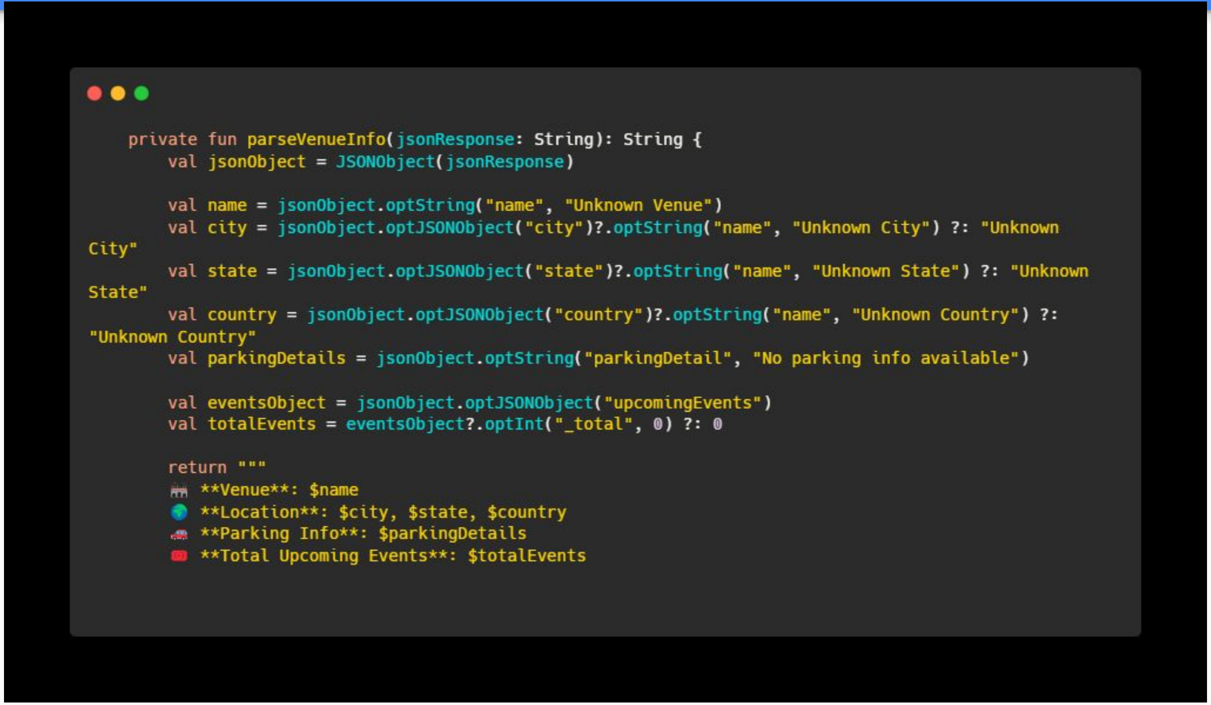
58 °F
clear sky

Wind: 3.6 mph
Clouds: 0%
N/A

58°F
Thu 05:00 PM

56.4
Thu 08:

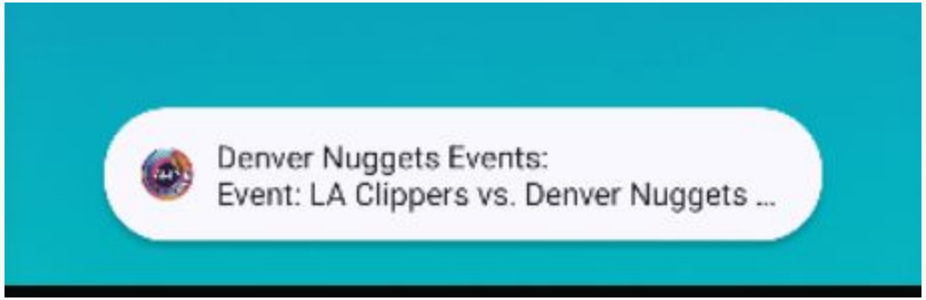
Live Ball Arena Updates

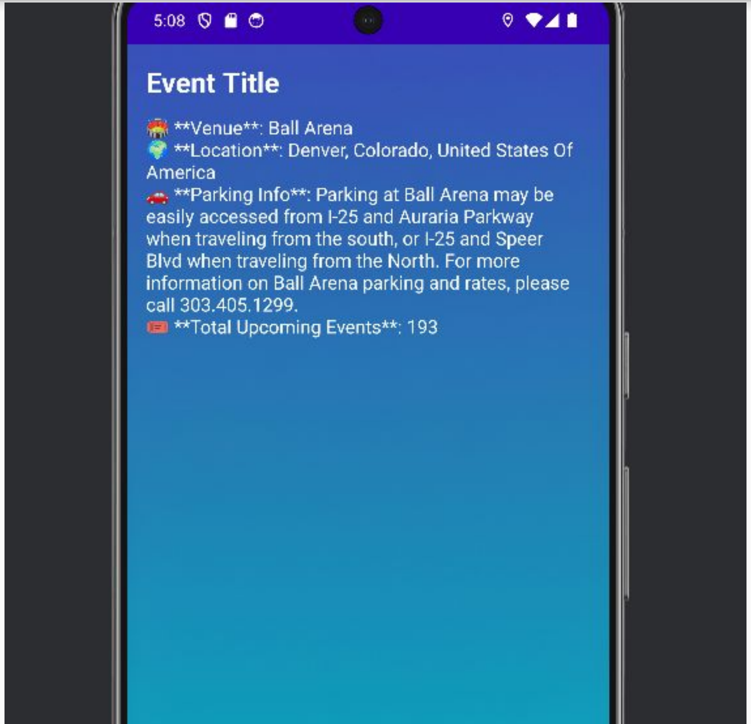


```
    private fun parseVenueInfo(jsonResponse: String): String {
        val jsonObject = JSONObject(jsonResponse)

        val name = jsonObject.optString("name", "Unknown Venue")
        val city = jsonObject.optJSONObject("city")?.optString("name", "Unknown City") ?: "Unknown
City"
        val state = jsonObject.optJSONObject("state")?.optString("name", "Unknown State") ?: "Unknown
State"
        val country = jsonObject.optJSONObject("country")?.optString("name", "Unknown Country") ?:
"Unknown Country"
        val parkingDetails = jsonObject.optString("parkingDetail", "No parking info available")

        val eventsObject = jsonObject.optJSONObject("upcomingEvents")
        val totalEvents = eventsObject?.optInt("_total", 0) ?: 0

        return """
        🏟 **Venue**: $name
        🌍 **Location**: $city, $state, $country
        🚗 **Parking Info**: $parkingDetails
        🎟 **Total Upcoming Events**: $totalEvents
```

Live Ball Arena Updates

- With our live Ball arena updates users will be able to avoid lots near the arena during events to minimize congestion.
- Visitors will be able to get event updates to coordinate parking efficiently or participate in arena activities.


Denver Nuggets Events:
Event: LA Clippers vs. Denver Nuggets ...


Event Title

🏟 **Venue**: Ball Arena
🌍 **Location**: Denver, Colorado, United States Of America
🚗 **Parking Info**: Parking at Ball Arena may be easily accessed from I-25 and Auraria Parkway when traveling from the south, or I-25 and Speer Blvd when traveling from the North. For more information on Ball Arena parking and rates, please call 303.405.1299.
🎟 **Total Upcoming Events**: 193

Live Demonstration

Acknowledgments

- API's
  - Openweather
  - Ticketmaster
  - Google maps SDK
  - Google AI studio
- Firebase
- ChatGPT
- Dr.Paul's guidance along the way.

The End