Ace Restaurant Android Application: Design Defense

Ivan Novasak

Southern New Hampshire University

IT 633: Mobile Application Development

Dr Venkatesh Baglodi

2025 March 9

Ace Restaurant Android Application: Design Defense

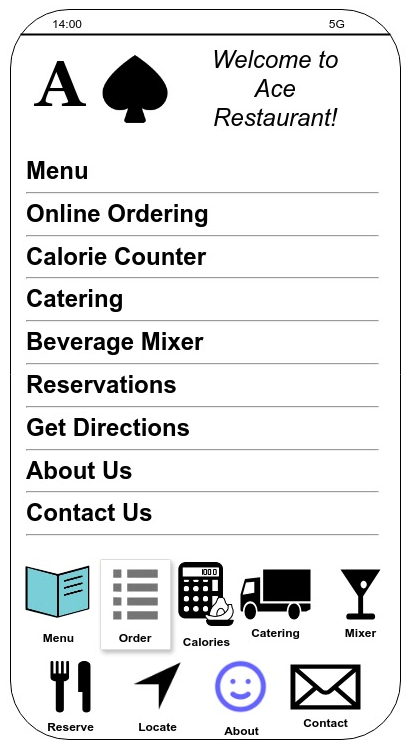
The purpose of this document is to state the functions, design decisions, and interface requirements for the Ace Restaurant Android application, and how these decisions address customers’ and the restaurant’s needs. Other areas to be covered are the completed app’s screenshots compared with an existing app, OpenTable, which has a similar purpose; a comparison between iOS and Android development; the development timeline; and compliance with privacy laws.

**Function and Requirements**

# Application Functionality

This application is comprised of three primary screens: a main menu which is what the user sees when they first open the app, a screen for the user to enter and confirm their address to get directions to the restaurant from, and a reservations screen which is where the customer can enter reservation details to be sent to the restaurant. Figures 1 through 3 show each screen’s expected design, with the main menu screen being based on a design given by Dr Venkatesh Baglodi and shown in (Baglodi, n.d., p 2). The author oversees programming the application’s main menu screen, directions entry screen, and reservation entry screen. For features not developed by the author, a message stating ‘To be developed later’ will be displayed should the user try to choose an option that is not available.

**Figure 1**

*Main menu screen design mock-up*

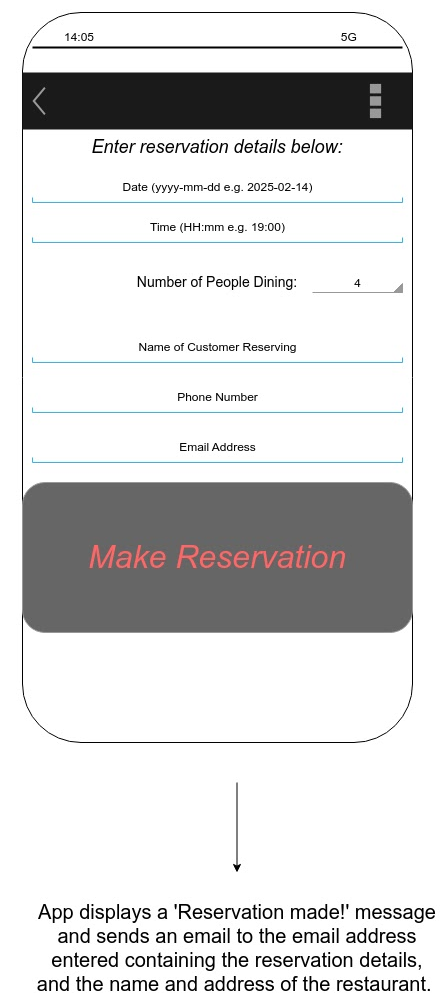
**Figure 2**

*Starting address entry screen mock-up*

**

**Figure 3**

*Reservation entry screen mock-up*

**

The main menu is the screen the user sees first and has a greeting with the restaurant’s name and logo up top, followed by a text menu with a listing of functions, and at the bottom a graphical listing of the same menu functions. When the user chooses either Reservations/Reserve or Get Directions/Locate, the app will load the relevant screen for the user to enter the origin address or their reservation details. As of creation of these mock-ups, the date/time was set as a text field, but in practice a date/time picker like what James C Sheusi describes in his book entitled *Android Application Development for Java Programmers* will be used to prevent erroneous date or time entries (Sheusi, 2013, pp 78–81). Red and green colours and italic text are used to make those areas of the screen stand out more for the customer to notice. These colour choices are for the buttons where the customer will submit their reservation entry or get directions.

**Application Fidelity**

The business requirements for the functions the author oversees were set out by Dr Venkatesh Baglodi in (Baglodi, n.d., p 2) and are as follows:

1. Use both swipe navigation (main menu) and icons like shown in their mock-up
2. Allow the user to get directions to the restaurant via the Google Maps API
3. Make an interactive reservation system that uses a variety of input methods.

The author’s user interface mock-ups accomplish these goals in that both a menu and icons are shown in the main menu screen, the directions entry screen is clear and has only what the user needs to enter to start getting directions, and the reservation screen uses text for contact information but a date/time picker for the reservation time. Large buttons with attractive colour italic text are used to confirm reservation and address entry. This is in line with what Greg Nudelman says in (Nudelman, 2013, pp 60/61) about persona and vision in that since the app will be used by customers who are on-the-go, it needs to be clear and have only the most necessary actions to function.

**Design Decisions**

The target audience for Ace Restaurant’s Android app was identified by the author in (Novasak, 2025, p 3) as:

* A smartphone or tablet user (getting directions from home or outdoors)
* Singles, couples, families, friend groups, or larger groups (reservations and catering)
* An adult will be who makes the reservation, given the possibility of alcohol consumption at the restaurant.

Another person will be handling the app’s food and beverage menu functionality, with the author overseeing the customer getting directions and making reservations - two areas covered by the target audience identification. The design decisions made need to allow the app to work on both smartphones and tablets and to allow for multiple people in a single reservation party. Text sizes are to be specified in Density-independent Pixels (dp) or Scale-independent Pixels (sp) units instead of pixels (px) or point (pt) format to ensure it looks the same on different device screen sizes (Sheusi, 2013, p 68). This will ensure that items will be placed in the same areas of the screen at the same proportions regardless of whether the device is a small smartphone, large smartphone, or tablet. Other design decisions to be made involve the change to a date/time picker instead of forcing the user to enter the date and time as text. Requiring text can cause the problem of the customer needing to know a specific format to enter the date and time in, whereas a date picker is easier to use when on the go. Another area to be considered is the number of customers. Originally, a Spinner (Sheusi, 2013, pp 74–77) was chosen as the entry method, which is an object the user can tap which will display a listing of valid values. Using one of these can prevent invalid entries but takes more time to choose an entry. It may be more logical to use a numeric text box and require a minimum and maximum number like 1 to 12.

**Interface Requirements**

The following elements are used in each screen:

* Main Menu screen:
  + Text listing of features
  + The same features listed as icons at the bottom
  + A greeting at the top
* Origin address entry screen:
  + Submit button labelled ‘Get Directions’ in green italics text
  + A text box for the customer to enter their starting address
* Reservation entry screen:
  + Text boxes for the name of the reserving customer, email address, and phone number
  + Numeric text box (or Spinner) for number of people dining with the reservation
  + Date picker
  + Time picker
  + A submit button labelled ‘Make Reservation’ in red italics text.

For the main menu, besides the specifications sheet by Dr Baglodi, it is typically good practice to use both text and icons so that customers can more easily identify the app’s features. The icons on the button may be easier to access with either hand. The date and time picker will ensure that any entered dates are valid and will not require manual adjustment due to formatting differences or invalid entries like 32nd December or 2-digit vs 4-digit years. Text boxes are needed for the customer’s name, email address, and phone number because these are entries that can have variable length. The buttons are designed to be large and attention-grabbing with red or green italics text to give a livelier interface element to the user. A choice of different colour for the directions button vs the make reservation button was decided so the customer will have less chance of accidentally entering the wrong submission.

**Mobile Application Framework**

**Framework Selection**

The framework chosen was for Ace Restaurant and the design requirements called for a multi-screen app with a menu and two chosen features amongst a selection of 10 choices (Baglodi, n.d., pp 2/3). The framework requires both icons and swipe navigation which was successfully included via the menu and two bottom rows of icons. Swipe navigation is automatically achieved in current Android versions where the back button is an inwards swipe from the side of the screen towards the centre. Specific items needed were buttons, text entry boxes, date selector, time selector, numeric entry, phone number entry, and email entry (the latter three are also text entry boxes, but with different keyboard types).

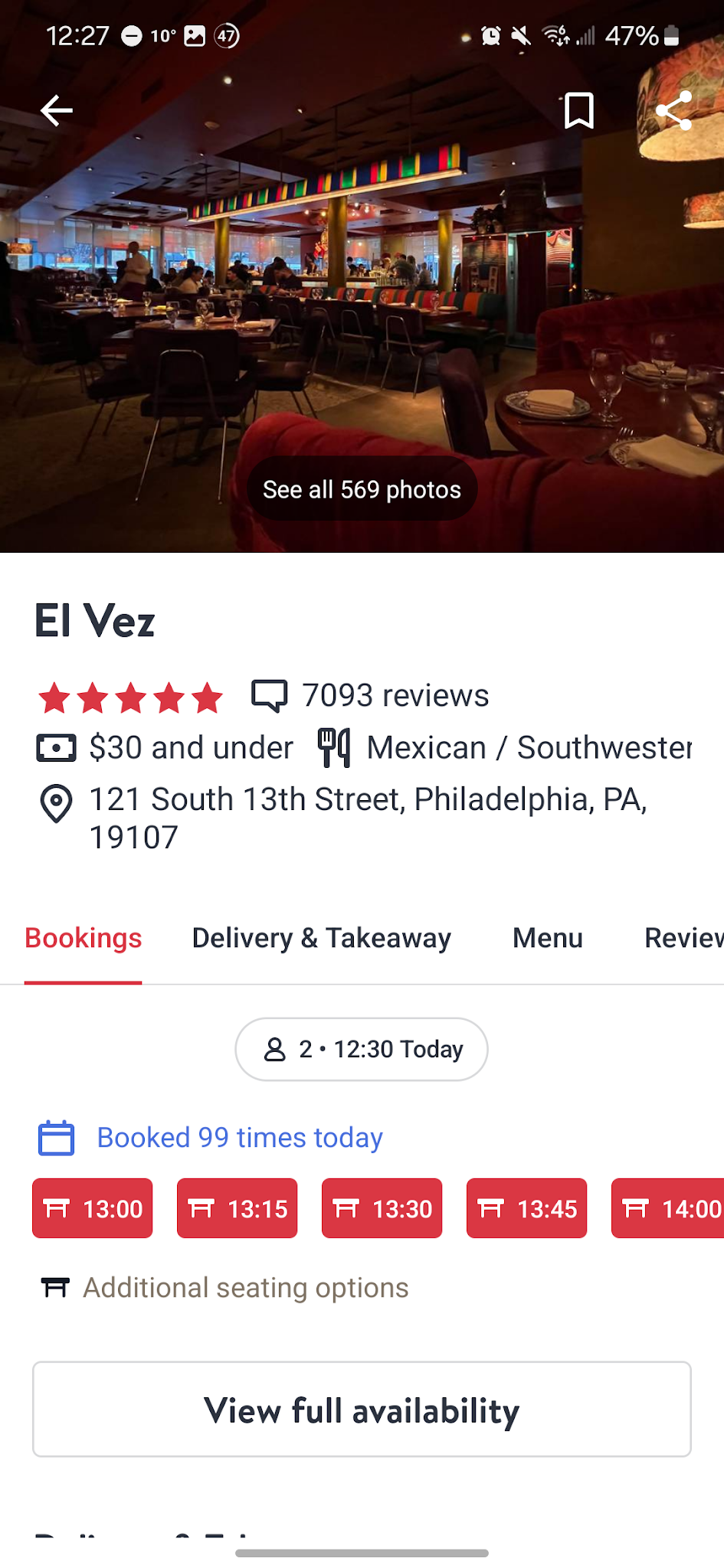
**Framework Evidence**

***OpenTable***

OpenTable (Google, n.d.) is a successful restaurant reservation app and website that allows customers to find restaurants and reserve tables at them. Figures 4 through 7 show how that app looks when viewing the El Vez restaurant in Center City, Philadelphia. This is an example showing that Ace Restaurant can incorporate similar designs for their app and the author’s contributions of the menu, address entry, and reservation entry are essential to any restaurant reservation app like OpenTable and Ace.

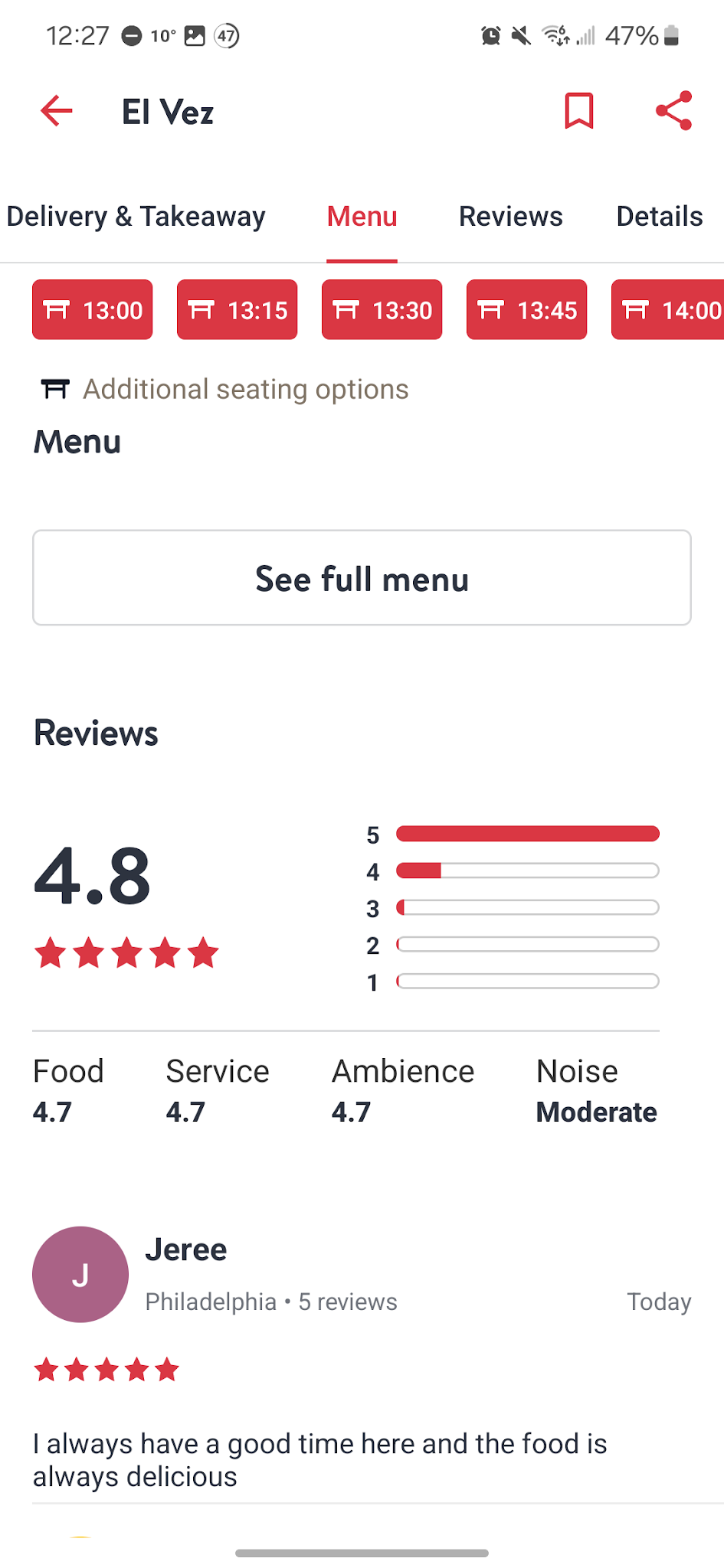
**Figure 4**

*OpenTable app screenshot showing rating, review count, address, next few available reservation times.*



**Figure 5**

*OpenTable app screenshot showing average rating, ratings for food/service/ambience/noise, and a written review.*

****

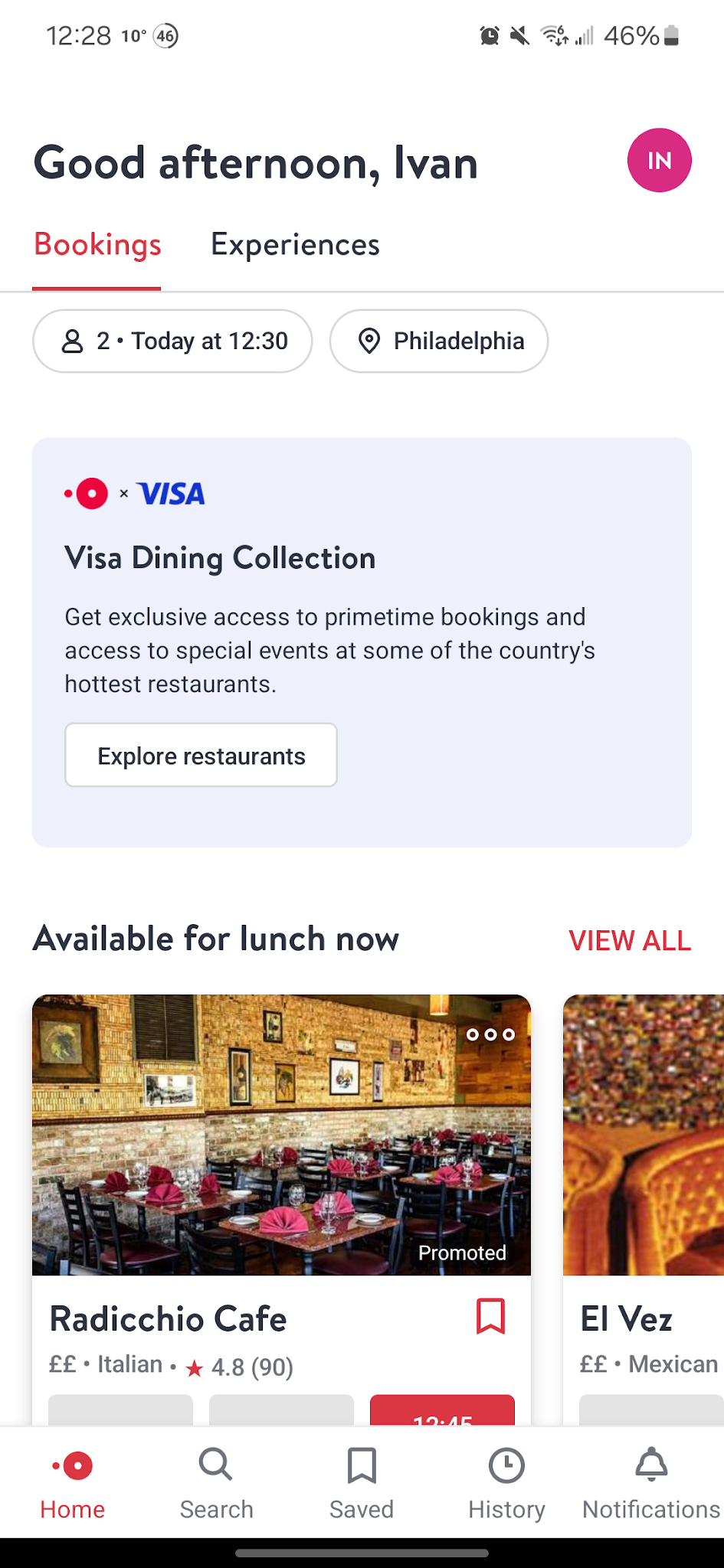
**Figure 6**

*OpenTable app screenshot showing El Vez’s location on a map. When the Directions button is clicked, the phone opens Google Maps with their location pre-printed as the destination*



**Figure 7**

*OpenTable app screenshot showing the initial screen upon opening the app.*

**

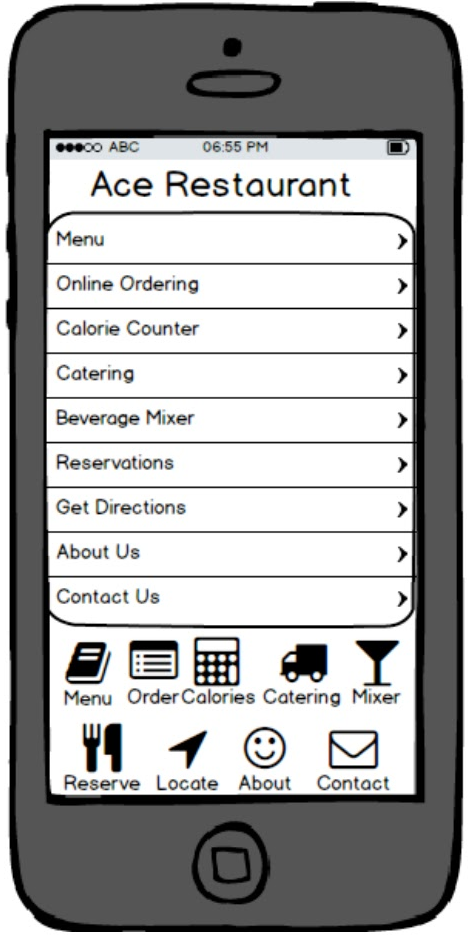
**Framework Alterations**

***iOS-style UI in the Requirements Yet Android is the Target OS***

Figures 8 and 9 compare the user interface differences between the original specification from Dr Baglodi, which appears more like an iOS design, and the implemented user interface, which is destined for Android.

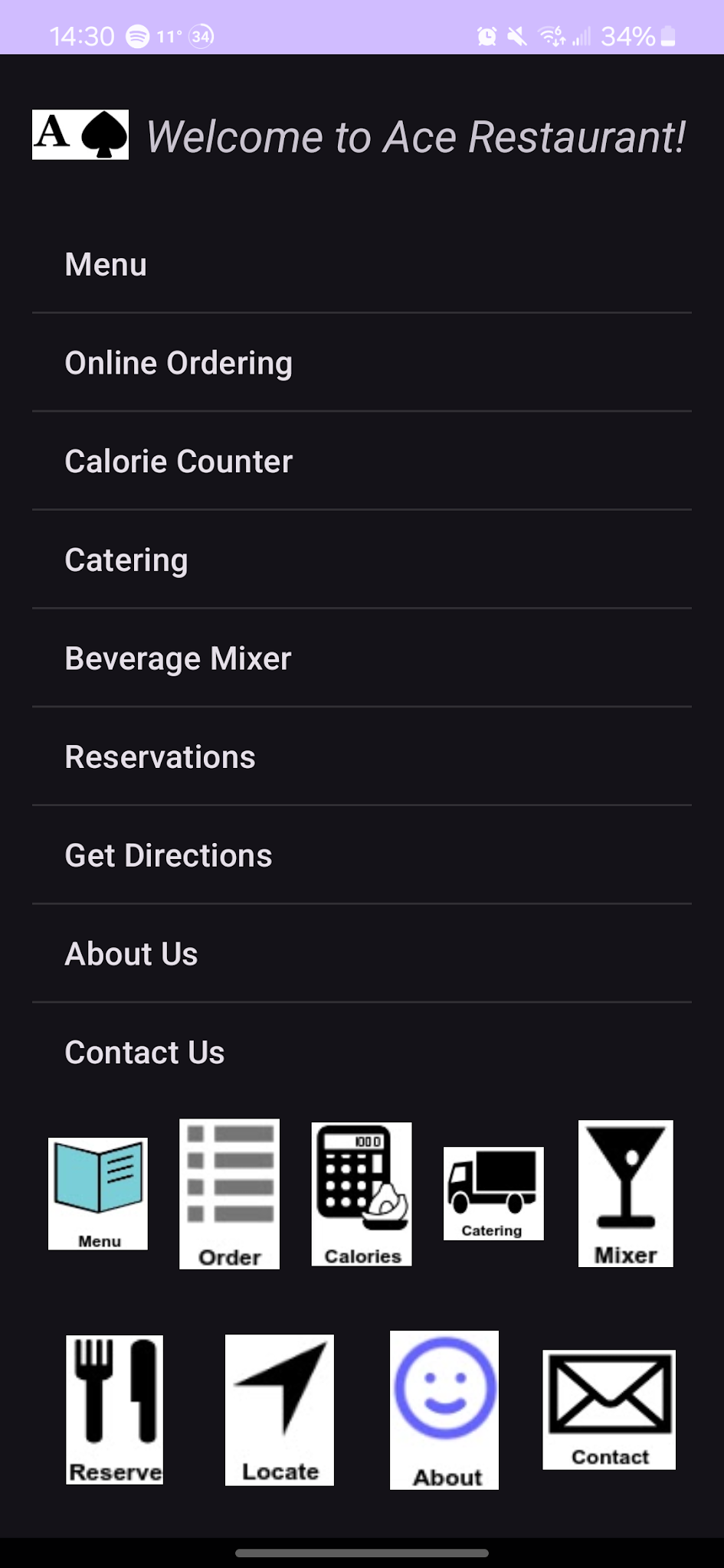
**Figure 8**

*Original sketch from Dr Baglodi’s requirements listing. Note the iOS style in the phone sketch and menu entries showing arrows on the sides.*



**Figure 9**

*Screenshot showing the implemented Android app running on a Samsung Galaxy S23 phone. The menu is shown, and the app automatically supports Dark Mode on phones.*



The modifications made were based on the examples Greg Nudelman showed in (Nudelman, 2013, pp 22/23). The menu was implemented without needing the arrows, so that the aesthetic is in line with Android interface requirements rather than iOS. A numeric text box entry was chosen instead of a wheel spinner for the number of customers reserving. The date picker with a calendar selector was chosen for the Android implementation rather than using spinner wheels for the year, month, and day of month, as the calendar allows the user to see the weekday labels and preprints the current date when it is initially selected. The confirmation buttons were chosen as larger buttons that take up most of the screen’s width for easier tapping when on-the-go, as opposed to smaller ones shown in (Nudelman, 2013, p 15) where they show a comparison of iOS and two different Android button styles. Figures 10 through 17 show the rest of the user interface with these changes in place.

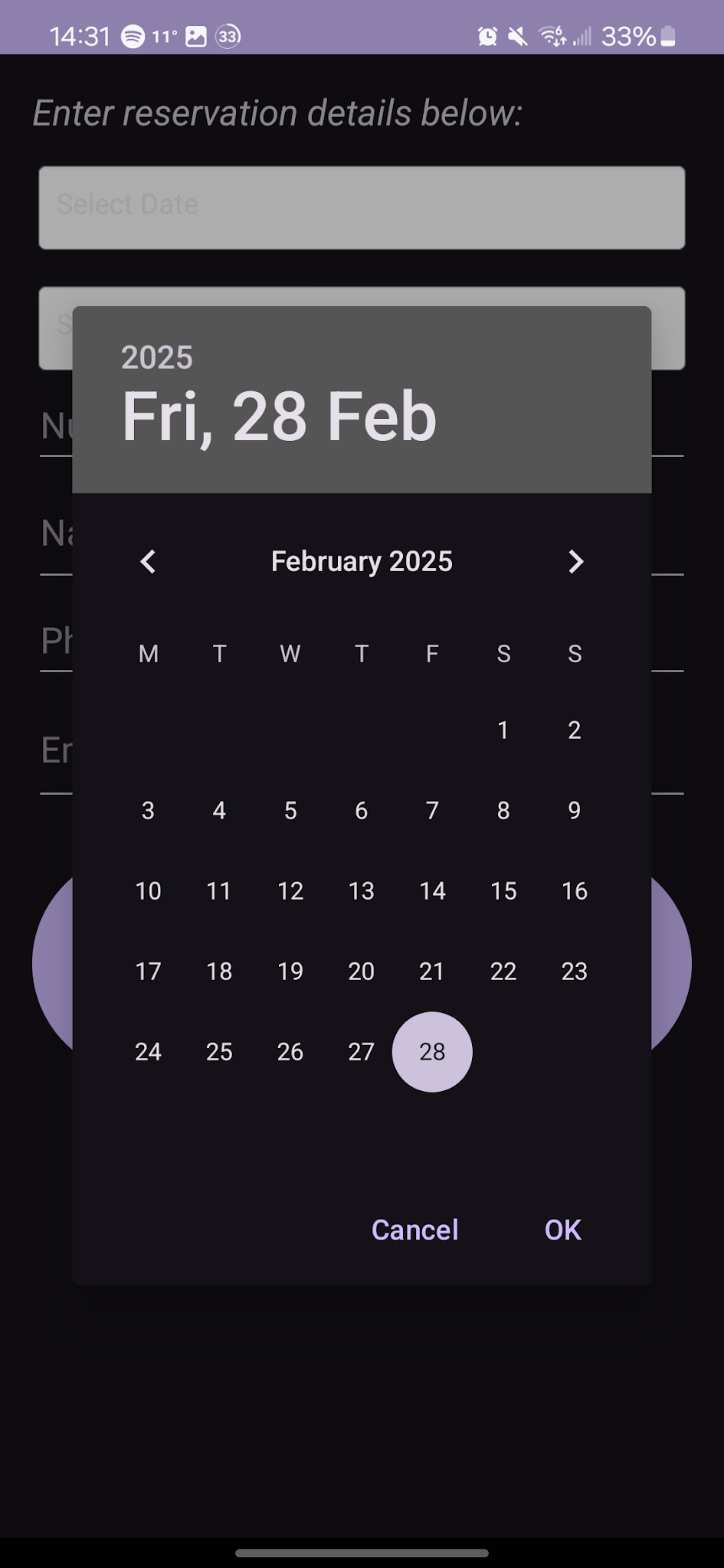
**Figure 10**

*Origin address entry screenshot*



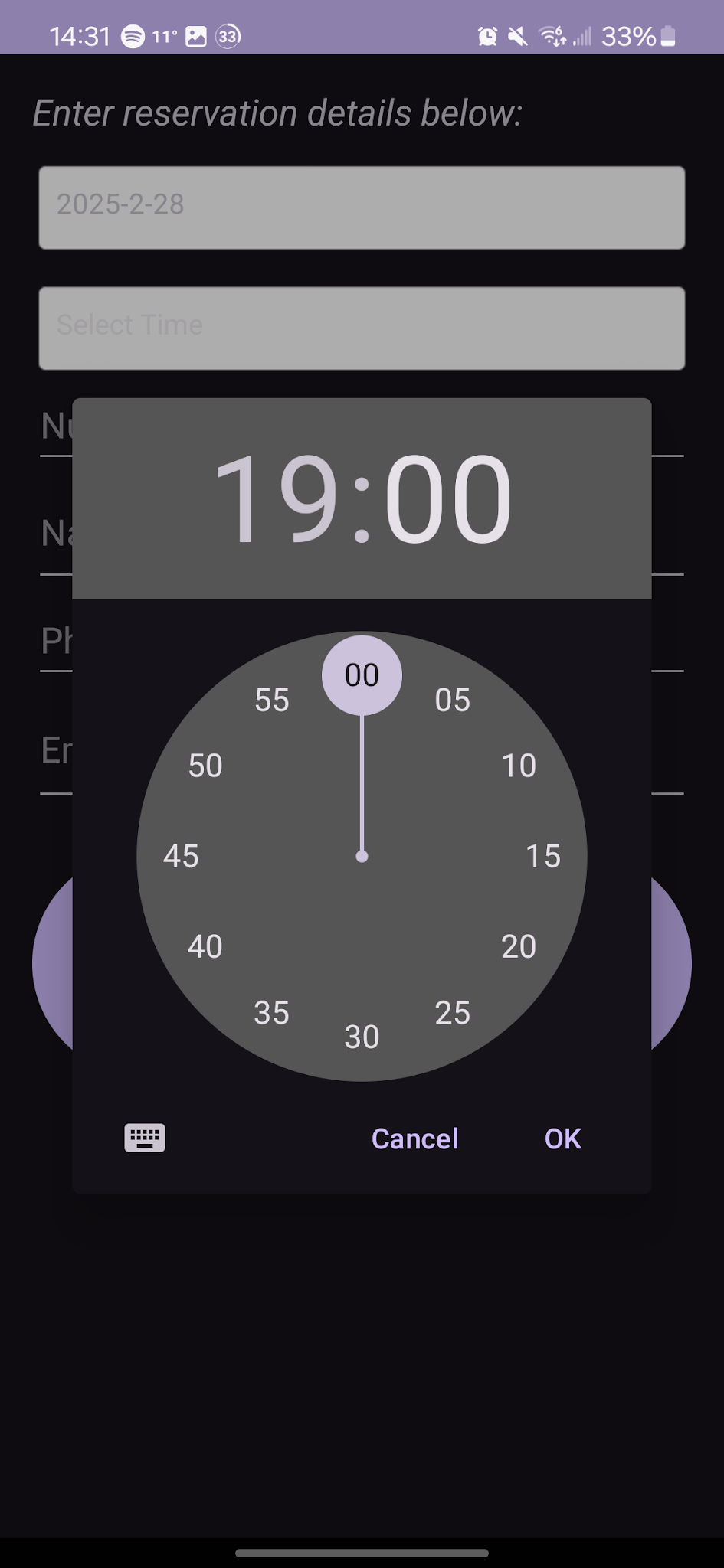
**Figure 11**

*Screenshot showing the date selector*



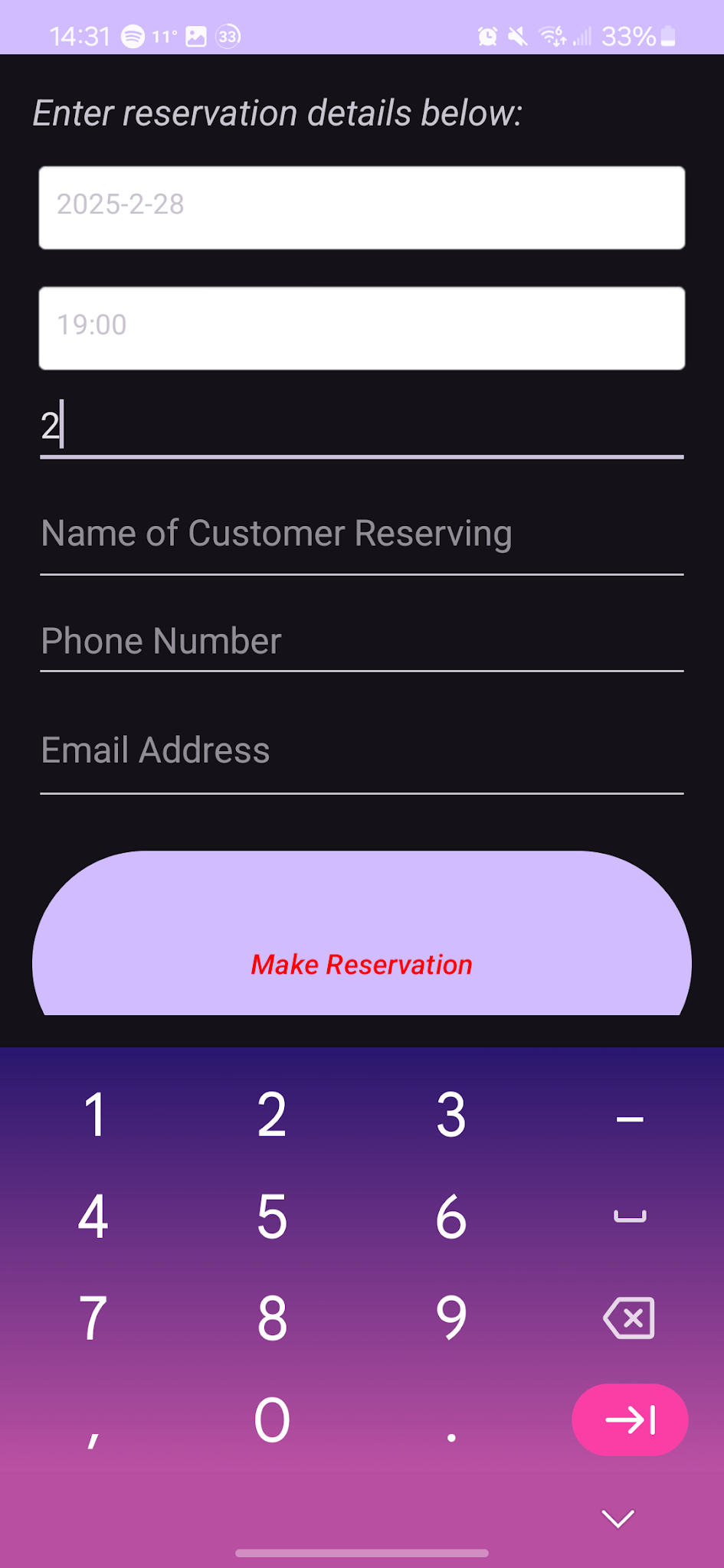
**Figure 12**

*Time selector screenshot*



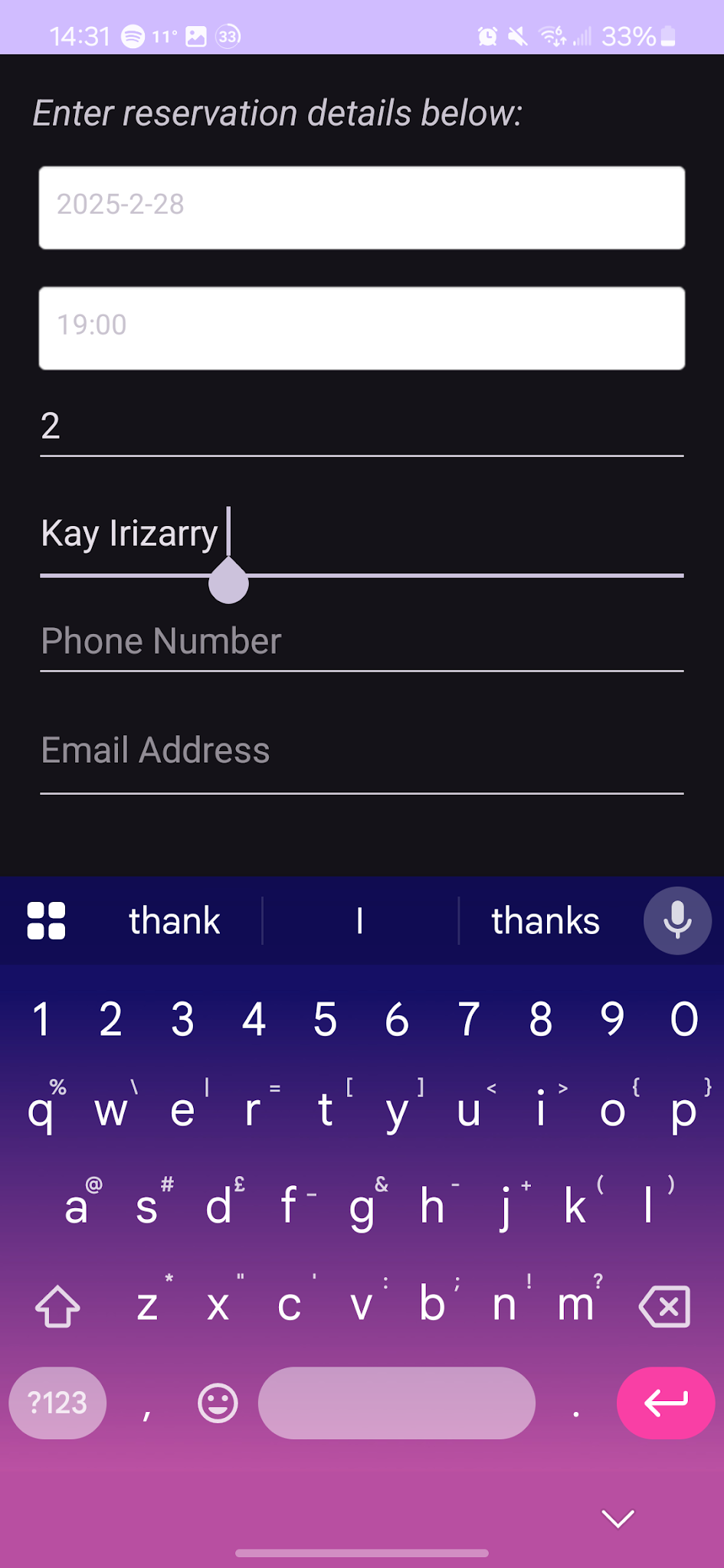
**Figure 12**

*Numeric entry screenshot*



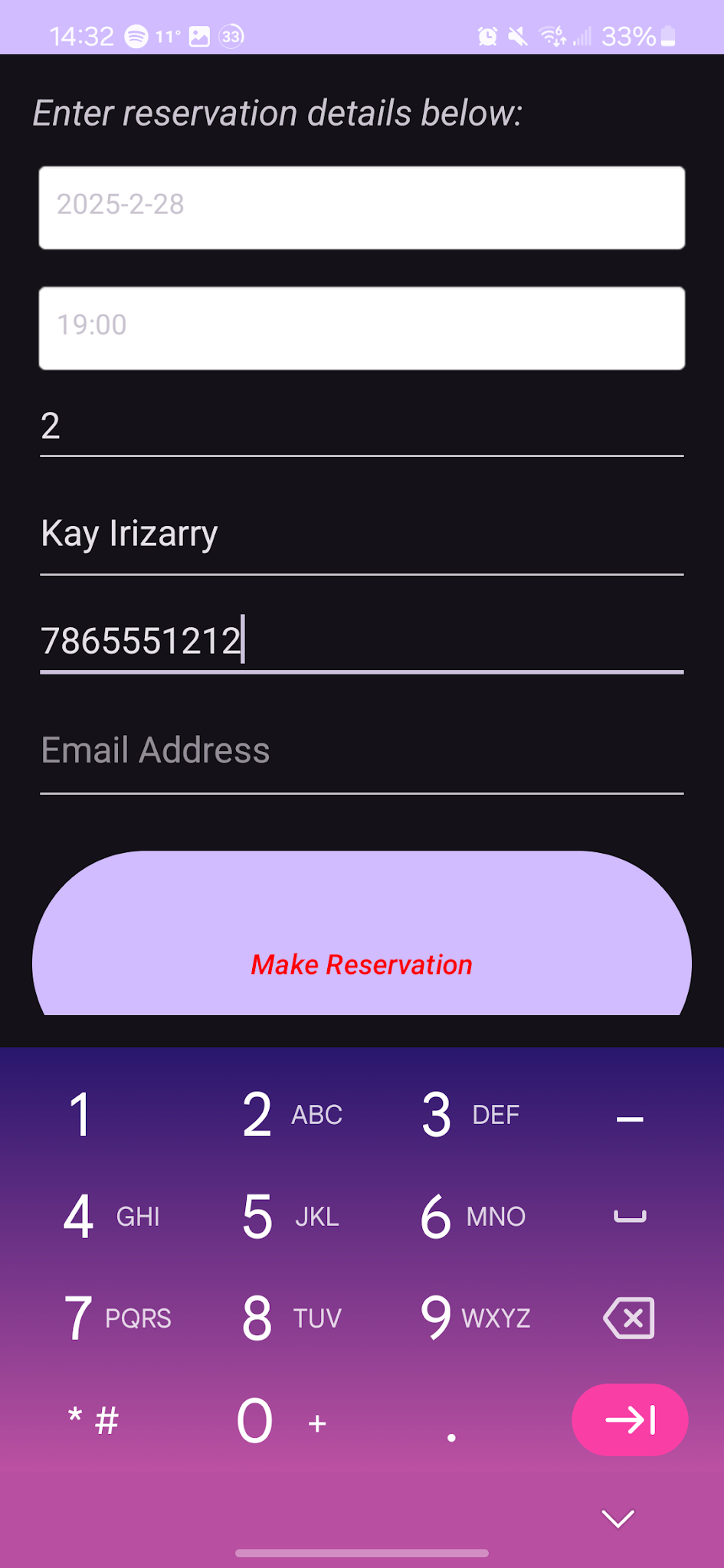
**Figure 13**

*Standard text entry screenshot for entering the customer’s name*



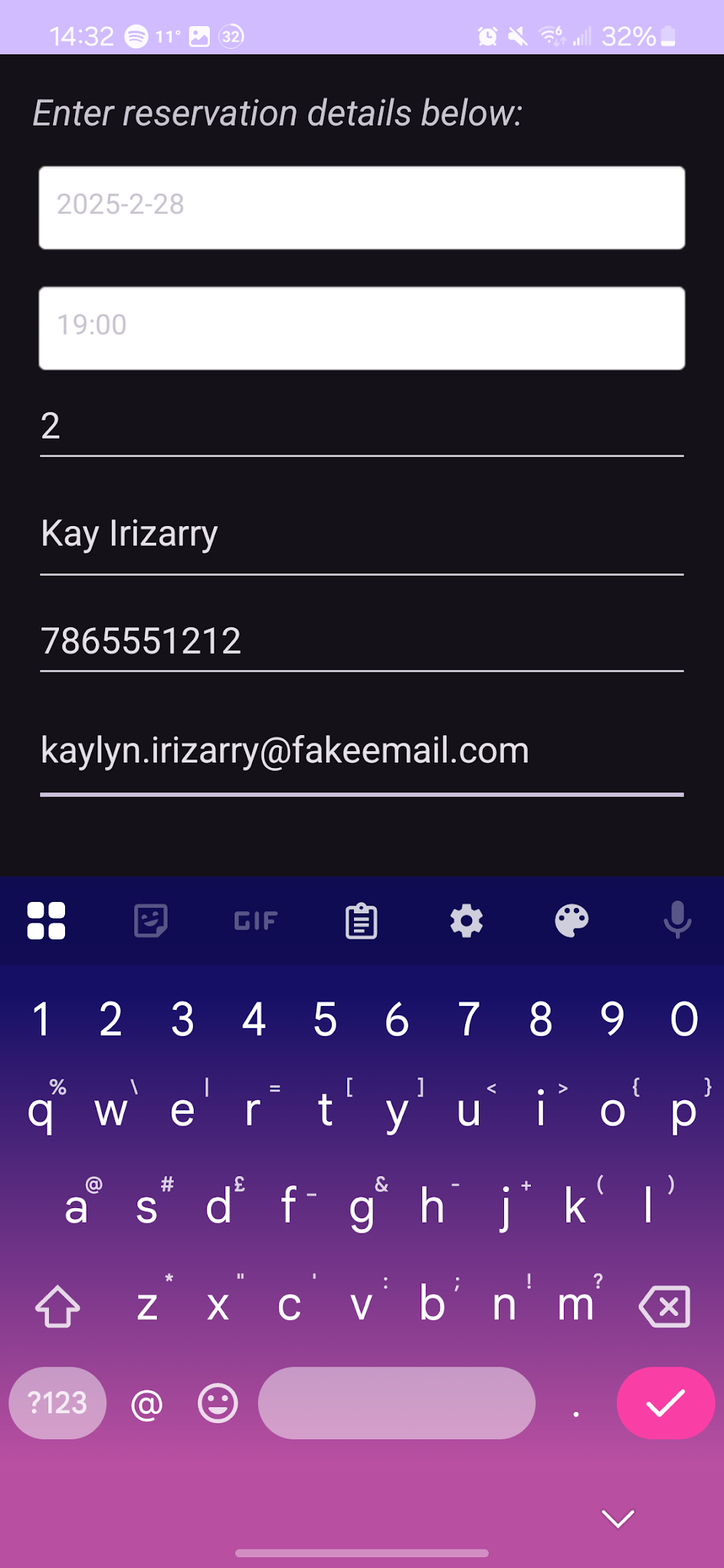
**Figure 14**

*Phone number entry screenshot. Notice the difference in the keyboard from the numeric entry.*

**

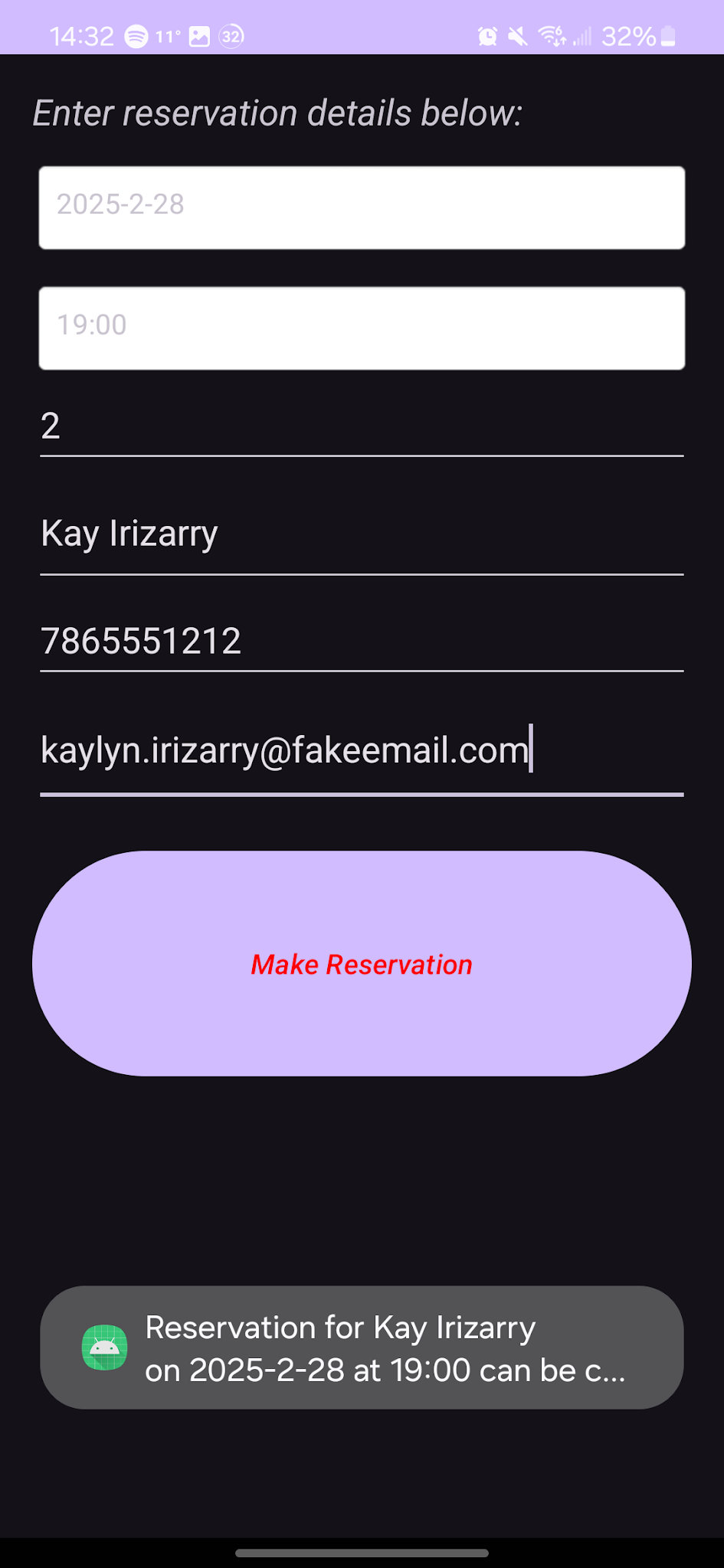
**Figure 15**

*Email address entry screenshot. Notice the slight difference in keyboard from the standard text entry for name entry.*

**

**Figure 16**

*Reservation confirmation screenshot*

**

**Figure 17**

*Placeholder screen for a feature that has not yet been developed*

**

**Mobile Application Approach**

**OS Comparison**

***Developing for Android vs iOS***

According to EGO Creative Innovations, five differences that arise between developing for Android vs iOS (the two major mobile operating systems that dominate the market lately) involve (EGO Creative Innovations, 2021):

1. Programming languages
2. Development environments: different integrated development environments (IDEs) in use
3. System-specific design differences
4. Development complexity.

For programming languages, Android Java and Kotlin, which are relatively open languages, one of which (Java) is heavily documented and has been around since the 1990s (EGO Creative Innovations, 2021). Apple’s iOS, on the other hand, uses a proprietary language called Swift (EGO Creative Innovations, 2021). Regarding understanding, both Kotlin (Android’s newer development language) and Swift are easy to learn and use, whereas Java, which is Android’s older, primary language, is more verbose and harder for a novice to get started with making apps (EGO Creative Innovations, 2021).

Both operating systems have their various development environments, which means using an IDE to code and test the apps (EGO Creative Innovations, 2021). For Android, Android Studio is typically used and is the environment officially recommended by Google (EGO Creative Innovations, 2021). The author has noticed that if one is used to the JetBrains IntelliJ IDE for coding in Java, Android Studio’s UI is very similar, so IntelliJ users will feel at ease. Apple’s iOS requires the use of their XCode IDE for all iOS app development (EGO Creative Innovations, 2021). Both IDEs have many features that help the user spot errors in code as well as the ability to test the code destined for different types of mobile devices within the IDE (EGO Creative Innovations, 2021).

There are some differences in developing for Android vs iOS. Android is more difficult to develop for due to a higher likelihood for errors as well as more fragmentation in device availability that need to be considered (EGO Creative Innovations, 2021). In Android, a typical app is made up of activities, one for each screen the app will have, and fragments, which are code portions that are parts of an individual screen and are used for different UI elements (EGO Creative Innovations, 2021). In contrast, iOS uses view controllers for both multi-screen as well as single-screen apps, and these can work with both portions of an app screen and a full screen (EGO Creative Innovations, 2021). With iOS, a developer can either code an app directly or make it in storyboard format, which could be a way to ease new developers in without requiring a lot of code knowledge at the outset (EGO Creative Innovations, 2021).

Android and iOS are very different when it comes to overall development complexity, because iOS has a limited number of devices, all from one company (Apple), and all in a rather narrow range of formats: small phones and larger tablets (EGO Creative Innovations, 2021). Android, on the other hand, has many different screen sizes and device shapes made across a variety of manufacturers, each of which with their own OS release schedule and differentiation tweaks to give some extra finishing touches on the OS (EGO Creative Innovations, 2021). This variety of different device sizes, manufacturers, and OS versions is collectively known as fragmentation, and developers need to rigorously test their app on a variety of device form-factors and different OS versions, depending on their market (EGO Creative Innovations, 2021).

***Ace Restaurant***

For developing the Ace Restaurant Android app, the author used Android Studio and did not need to use fragments within this app, as the features being developed were not as complicated to require fragments; just using separate screens worked well. Each screen was made for a single purpose (main menu, starting address entry, and reservation entry). Testing was done in both Android Studio’s device simulator and on the author’s Samsung Galaxy S23 smartphone, where it was confirmed that the app was performing well.

**Development Approach**

The full timeline of the development is as follows in Table 1. Milestone entries in the tables are in *italics*.

**Table 1**

*Development timeline*

|  |  |
| --- | --- |
| **Date (yyyy-mm-dd)** | **Item Completed** |
| *2025-01-12* | *Chose Ace Restaurant for this Android app project* |
| 2025-01-18 | Defined the full app’s scope |
| *2025-01-19* | *Drew mockup user interface* |
| 2025-01-24 | Began writing alpha build code |
| 2025-01-25 | Wrote app pseudocode |
| *2025-01-28/29* | *Wrote more app code and began testing;*  *Encountered crashing problem upon user tapping menu options* |
| 2025-02-01 | Wrote document containing app’s function requirements, design decisions, interface requirements |
| *2025-02-02* | *Fixed crashing problem via adding Activity Manifest (Sheusi, 2013, pp 143/151) notations: beta release* |
| 2025-02-07 | Added bottom row icons |
| 2025-02-16 | Defined remaining requirements in journal document |
| 2025-02-17 | Fixed app bottom rows icons, added top logo |
| 2025-02-17 | Final testing of app performance and functionality on the author’s Samsung Galaxy S23 |
| *2025-02-18* | *Emailed app APK and source code files to Dr Baglodi for final review* |
| 2025-02-19 | Dr Baglodi emailed to say app is good and clear for release |
| *2025-02-22* | *Submitted final application and source code* |

**Approach Defense**

***Mobile Development Life Cycle (MDLC)***

According to Kacper Rafalski of Netguru, the mobile development life cycle is made up of six major stages that must take place for an app to be successfully deployed and maintained (Rafalski, 2025). They are as follows (Rafalski, 2025):

1. Planning
2. Design
3. Development
4. Testing
5. Deployment
6. Maintenance.

The first stage, planning, is where the developers decide on an idea for this app, research potential target audience, which developers oversee each portion of development, what platform(s) it will run on, tentative timeline, rough sketches of what the app will look like, and a vision of what success looks like (Rafalski, 2025). The second stage, design, is when the user interface is designed and the rest of the app’s overall look is decided (Rafalski, 2025). Prototypes of the design using wire frames are made at this point (Rafalski, 2025).

Now that planning and design are finished, the third stage can commence: development (Rafalski, 2025). This is where the app’s interface (often called front-end) and functionality (usually called back-end) are programmed in whatever languages are needed for the operating system or hardware in question, then the code can be tested in the form of alpha and beta prototypes (Rafalski, 2025). In this stage, the end-users’ hardware, different screen sizes and internet connectivity needs must be accounted for (Rafalski, 2025). After this comes the fourth development stage: testing (Rafalski, 2025). In an iterative development plan, this phase would be taking place during development, but it can also take place afterwards (Rafalski, 2025). The three main types of testing are a) usability, which is where the app’s basic functions are tested and ensure that the app is easy to use, where feedback can be gathered for improvement; b) performance testing, where speed and battery life when using the app as well as network performance on different networks is tested; finally c) quality assurance (QA) testing, which is where developers check compatibility with the different types of devices and OS versions, app stores, check if all functions of the app perform correctly, and ensure there are no privacy violations (Rafalski, 2025). If the QA testing fails, the app gets reported back to the main developers with the areas that need repair (Rafalski, 2025).

After development and testing comes deployment and maintenance, which are the fifth and sixth stages of mobile app development. Deployment is where the app is formally launched into the various app stores that the hardware supports, such as Google Play and Samsung Galaxy Store, and then users can submit reviews of the app (Rafalski, 2025). These reviews need to be monitored for any complaints on functionality or other problems the app is having that were unforeseen during development, then forwarded to the correct people on the team who can repair them (Rafalski, 2025). App maintenance is last and ongoing, where new features are proposed, either accepted or not, then if accepted, developed, and tested for a new version release after beta testing amongst a smaller group of users (Rafalski, 2025). For the Ace Restaurant app, the first five stages were used, with no subsequent releases expected.

**Industry Ethics**

***Privacy Concerns***

According to Jugnen Pandya, the Vice President of Expert App Devs, several privacy and consumer protection laws around the world that concern mobile app development are (Pandya, 2024):

* Personal Data Protection Bill (India)
* Digital Markets Act (Europe)
* California Consumer Privacy Act (USA)
* General Data Protection Regulation (European Union)
* Asia Pacific Privacy Regulations
* Data Protection Directive (European Union).

These laws concern both the capture and use of consumers’ data and specify what companies are allowed to do with their data. Consumers must be informed about what data is being collected and why and give users the option to download their data and delete it from the company’s server, specifically with the GDPR, which is a wide-ranging law (Pandya, 2024). During development, the company developing the app should contact a data privacy or cybersecurity lawyer, or a company like Expert App Devs. They will review the app’s specifications and the types of data that are needed from consumers and ensure that users are being given the option to consent to any required collection of their personally identifiable information (Pandya, 2024). All data collected should be encrypted and backed up in a place that is secure from cyber and real-world criminals. Training for staff on developing with the user’s privacy and data security first in mind, and compliance with the data protection laws is vital (Pandya, 2024). For the Ace Restaurant app, no personal data is being collected or stored, and is even automatically cleared when the user leaves the screen, they are on. Another aspect of these laws is they typically require the app to collect only the minimum data required (Pandya, 2024).

For the comparison between the GDPR and DMA, which are both used in the EU, these laws concern computer and mobile data but in different ways. The GDPR is about how a user’s data is collected and how their privacy and security are maintained; the DMA concerns which app stores apps are sold at and has the objective of eliminating gatekeeping and monopolisation by ensuring that apps can be sold in multiple app stores and users can have alternative stores loaded on their phones and tablets (Pandya, 2024).

# References

Baglodi, V. (n.d.). *IT 633: Mobile Application Frameworks and Functionality Sets*. Southern New Hampshire University. <https://snhu-media.snhu.edu/files/course_repository/graduate/it/it633/it633_mobile_application_framework.pdf>

EGO Creative Innovations. (2021, March 30). *5 Major differences between iOS and Android app development*. EGO Creative Innovations. <https://www.ego-cms.com/post/5-major-differences-between-ios-and-android-app-development>

Google. (n.d.). *OpenTable*. Google Play. <https://play.google.com/store/apps/details?id=com.opentable&hl=en_CA>

Novasak, I. (2025, January 19). *2-1 Assignment: Project Scope and Pseudocode*. Southern New Hampshire University.

Nudelman, G. (2013). *Android Design Patterns: Interaction Design Solutions for Developers*. John Wiley & Sons.

Pandya, J. (2024, December 11). *Privacy and Data Protection Laws for Mobile App Development*. Expert App Devs. <https://www.expertappdevs.com/blog/mobile-app-security-privacy-and-data-protection-laws>

Rafalski, K. (2025, March 1). *Mobile Development Lifecycle: Key Stages for Efficient App Creation*. Netguru. <https://www.netguru.com/blog/mobile-development-lifecycle>

Sheusi, J. C. (2013). *Android Application Development for Java Programmers*. Course Technology (Cengage Learning).