Panther Shuttle App

User and Developer Manual

Senior Design Project

Joseph Hilte, Tony Arrington, Jonathan Suo, Chase Monigle

CSE 4102 Computer Science Project 2

Dr Philip Chan

Table of Contents

1. Introduction 3
2. System Overview 3
3. User Manual 4
  3.1 Student Manual 4
  3.2 Driver Manual 9
  3.3 Manager Manual 12
4. Developer Manual 14
  4.1 System Architecture 14
  4.2 Technologies Used 15
  4.3 Database Design 15
  4.4 Source File Descriptions 16
  4.5 Important Methods and Functions 17
  4.6 Build and Deployment 17
  4.7 Testing and Troubleshooting 18
5. Future Work 18
6. Conclusion 18
Appendix A. Role Based Usage Examples 19

1. Introduction

The Panther Shuttle App is an Android-based mobile application developed to improve the on-campus shuttle experience for students, drivers, and managers. The application provides live shuttle visibility, daily schedule access, driver notifications, favorite-stop management, and a centralized manager portal for maintaining route stops and official schedules. By placing all major route information inside one mobile system, the project reduces confusion, improves communication, and helps users interact with the shuttle service more efficiently.

This manual combines a user manual and a developer manual into one document. The user portion explains how each role can use the application in practice. The developer portion explains the system architecture, database structure, major source files, and the methods used to implement the application. This format is intended to support both non-technical readers and technical reviewers such as instructors or future developers.

The overall design goal of the project was to build a practical campus transportation system rather than a simple class prototype with isolated screens. To support that goal, the app was structured around three distinct user roles. The student side focuses on convenience and visibility, the driver side focuses on communication and shuttle operation, and the manager side focuses on maintaining the official route data. All three roles are connected through Firebase so changes made by one authorized role can be reflected throughout the rest of the system.

2. System Overview

The Panther Shuttle App supports three user roles: Student, Driver, and Manager. Each role has a separate interface with features designed around that user's needs. Students are end users of the shuttle system. Drivers operate the route and communicate with students. Managers maintain stop locations and the official shuttle schedule. The application uses Firebase Authentication, Cloud Firestore, and Google Maps to support real-time updates and shared route information.

- Student role: view live shuttle location, daily schedule, notifications, favorite stops, estimated student demand, and next-stop ETA.
- Driver role: share live location, view next stop, estimate student demand, and send notifications to all students or targeted stop/time groups.
- Manager role: add, edit, and delete route markers, and create or update the official shuttle schedule by day of the week.

The manager side acts as the source of truth for stops and schedules. This means that the stop markers and route schedule viewed on the student and driver sides are pulled from the data entered by the manager. This design prevents the three interfaces from drifting out of sync and ensures that all users are working from the same route data.

3. User Manual

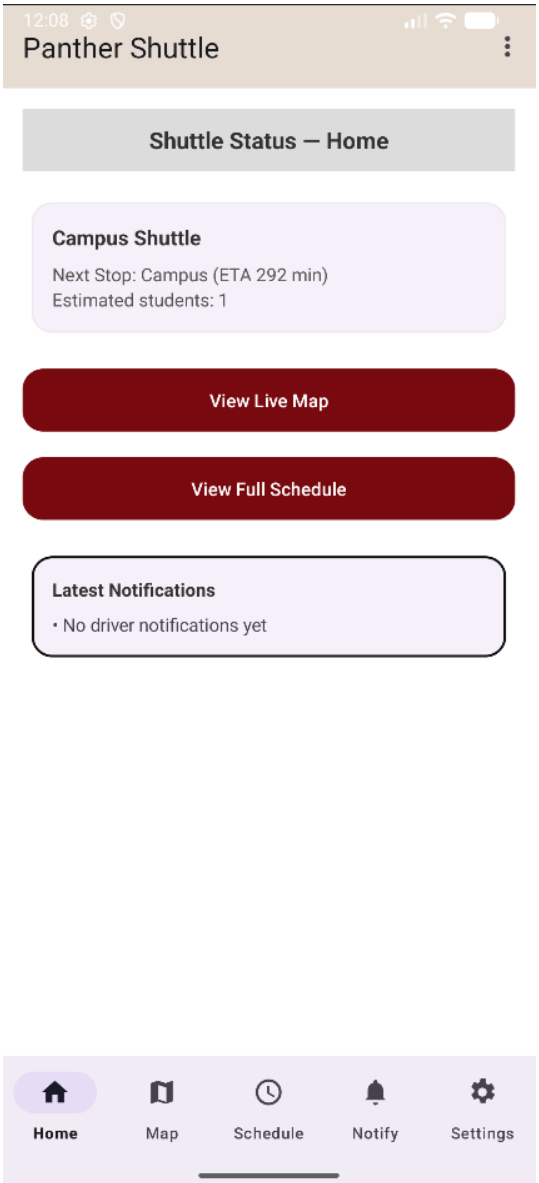
This section explains how to use the application from each role's point of view. Each role description includes the main features, screen behaviors, and an example usage scenario. When inserting screenshots into the final document, place them immediately after the subsection that describes the matching feature so the document is easy to follow.

3.1 Student Manual

The student side of the app is designed to help passengers quickly understand where the shuttle is, what the schedule looks like for the current day, whether there are any recent driver updates, and when to leave for a favorite stop. The student interface includes five main areas: Home, Map, Schedule, Notifications, and Settings.

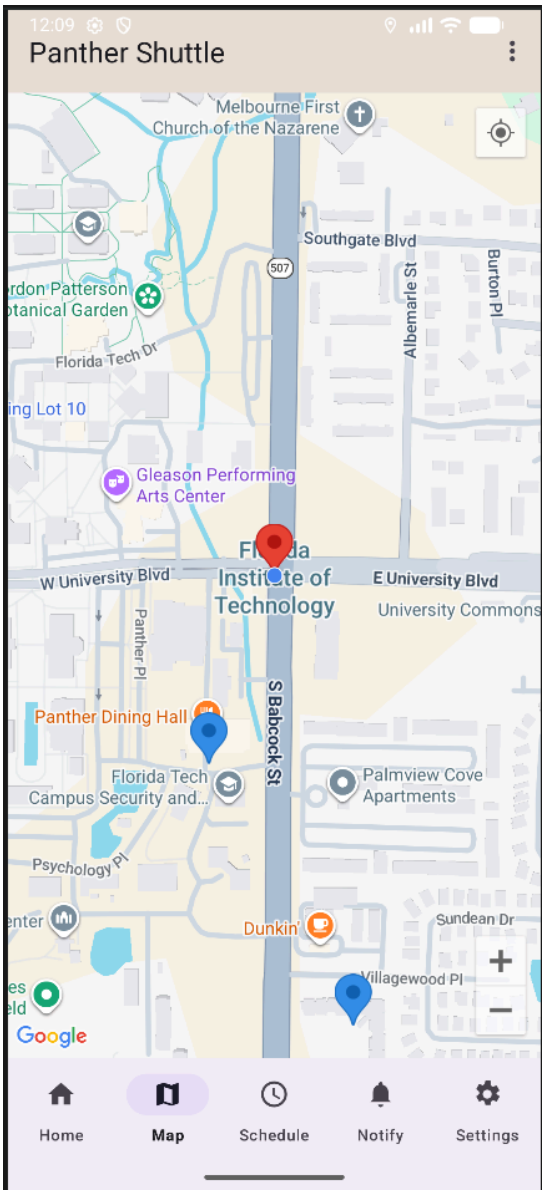
3.1.1 Student Home Screen

The Home screen acts as the student dashboard. It summarizes the most important shuttle information on one page. The screen displays the next scheduled stop for the current day, the estimated time remaining until that stop, and the estimated number of students expected at that stop based on favorite-stop data. The Home screen also displays the latest driver notifications that apply to that student. If a notification is dismissed on the Notifications screen, it no longer appears in the latest notifications box on the Home screen.



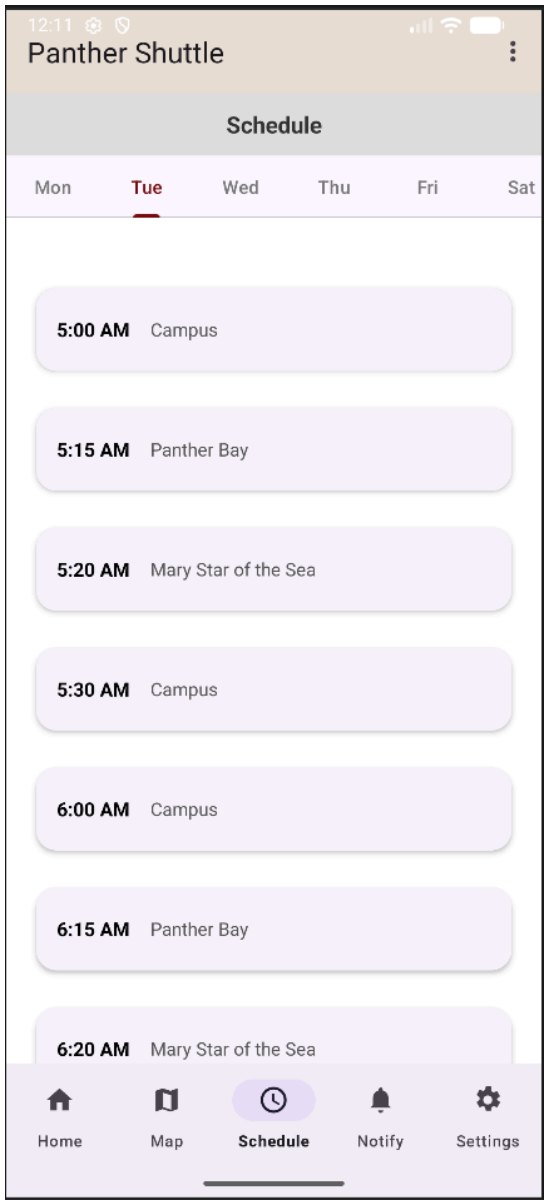
3.1.2 Student Map Screen

The Map screen allows students to visualize the current route. Shuttle stops created by the manager appear as map markers, and the driver's live location appears when the driver has enabled live location sharing. This helps students decide when to leave their current location and head toward the stop. If the driver has turned off live sharing, the route markers still appear, but the live shuttle marker is not shown.



3.1.3 Student Schedule Screen

The Schedule screen displays the manager-defined schedule by day of the week. The student can use the tabs to switch between Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, and Sunday. Each tab shows the stop names and times scheduled for that day. Because this screen is connected to Firebase, any updates made on the manager side automatically appear here once the data syncs.

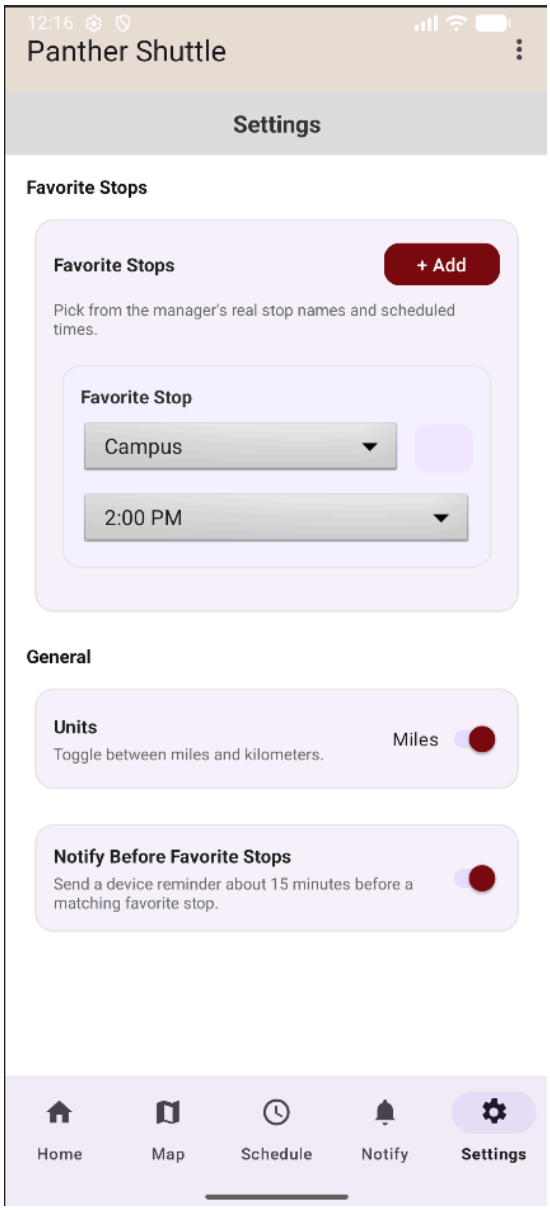


3.1.4 Student Notifications Screen

The Notifications screen displays driver messages sent through the driver interface. Messages can be broadcast to all students, targeted to a specific stop, or targeted to a specific stop and time. The app checks the student's saved favorite stops and only displays targeted notifications that match that student's profile. Students can dismiss notifications they no longer want to see. Dismissed notifications are hidden from both the notification list and the latest notifications box on the Home page.

3.1.5 Student Settings Screen

The Settings screen allows the student to manage favorite stops and preferences. Favorite stops are not arbitrary entries; instead, the student selects from the real stop names and real scheduled times defined by the manager schedule. This ensures the favorite-stop system is always aligned with the official route. The settings page also includes a toggle for receiving a reminder approximately 15 minutes before a scheduled favorite stop. This reminder is delivered as a local device notification while the app is active.



3.1.6 Student Example Workflow

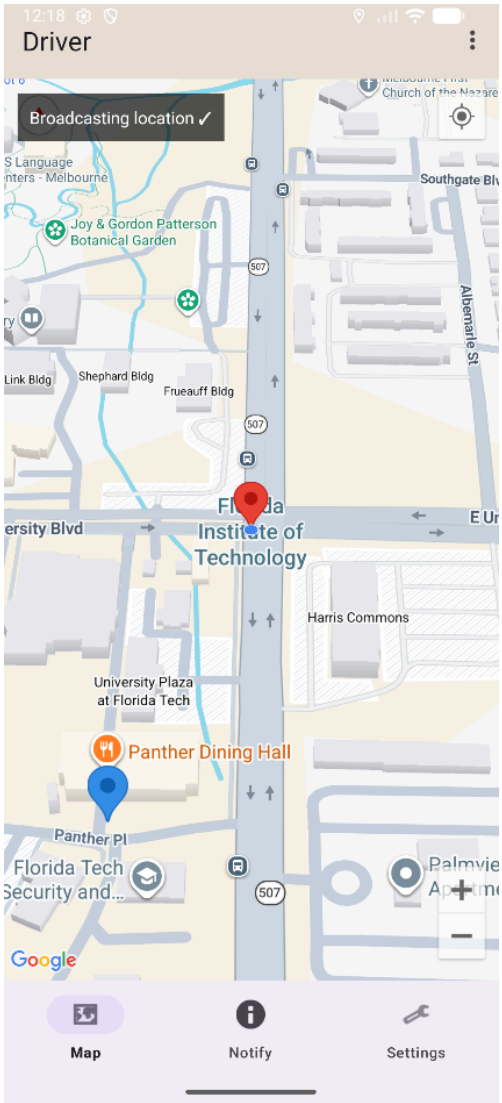
Example workflow: A student opens the app on Friday evening and checks the Schedule tab. The student sees that Panther Bay is scheduled for 8:15 PM and saves Panther Bay at 8:15 PM as a favorite stop in Settings. Later, the student returns to the Home screen and sees that the next stop is Panther Bay with an ETA of 12 minutes and an estimated student count of 4. A few minutes later, the driver sends a targeted notification for Panther Bay at 8:15 PM. The student sees this message in the Notifications tab and on the Home screen. If the reminder toggle is enabled, the app also issues a local reminder roughly 15 minutes before the stop.

3.2 Driver Manual

The driver side of the application is designed to support shuttle operation and student communication. The driver interface includes three major areas: Map, Notify, and Settings. These tools allow the driver to share location, monitor the next stop, and send messages to the appropriate group of students.

3.2.1 Driver Map Screen

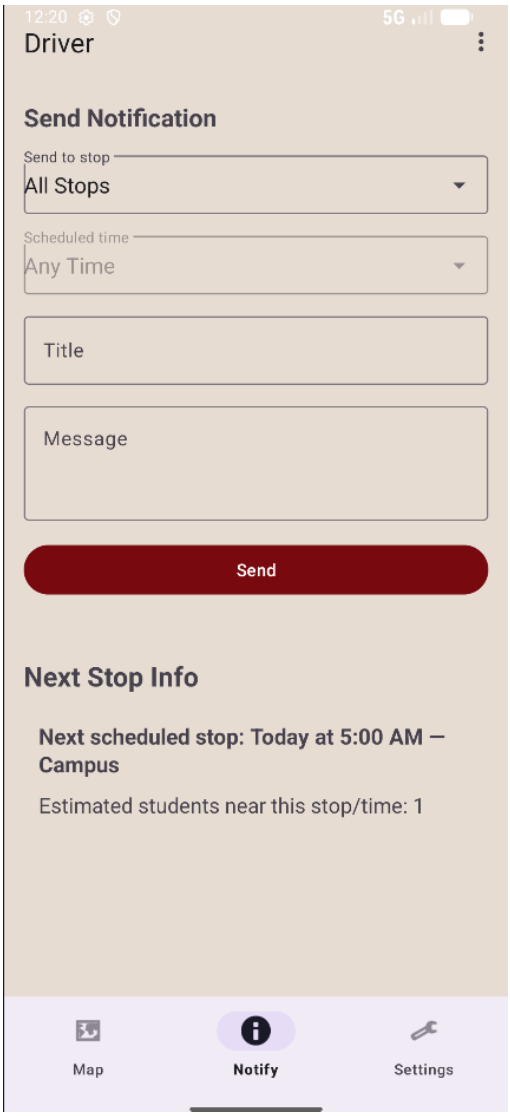
The Driver Map screen shows the driver's own current location and the shuttle stop markers maintained by the manager. If the Share Live Location setting is turned on, the app broadcasts the driver's current coordinates to Firebase on a timed interval. Students then see this live marker on their map. If the setting is turned off, the driver still sees the map for personal use, but the location is no longer shared with students and the old live marker is removed from Firebase.



3.2.2 Driver Notification Screen

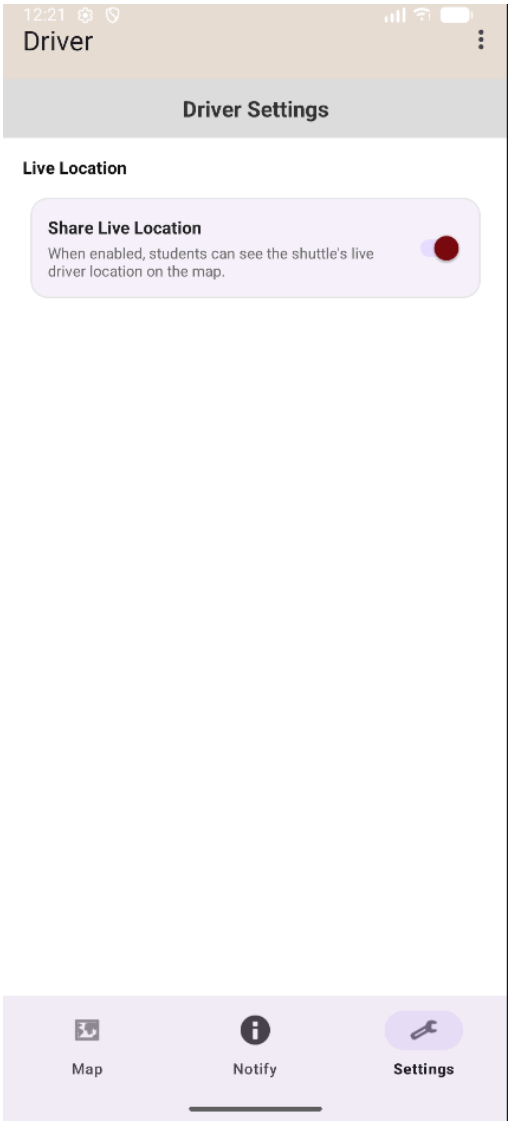
The Driver Notify screen is used to communicate with students. The driver can choose to send a message to all students, to a specific stop, or to a specific stop at a specific scheduled time. The stop list and time list on this screen come from the manager's official schedule, so the driver is always working with valid route data. The screen also displays the next scheduled stop and the estimated number of students expected at that stop, based on favorite-stop data entered by students.

This design helps the driver send more precise updates. For example, if the driver is approaching Panther Bay at 8:15 PM, a message can be sent only to students whose favorite stop matches Panther Bay at that time. This reduces irrelevant messages and improves communication quality.



3.2.3 Driver Settings Screen

The Driver Settings screen currently includes the Share Live Location toggle. This setting determines whether the driver's device publishes live location updates to Firebase. When the setting is enabled, the app displays a broadcast status message indicating that live location is being shared. When disabled, broadcasting stops and students can no longer see the live driver marker.



3.2.4 Driver Example Workflow

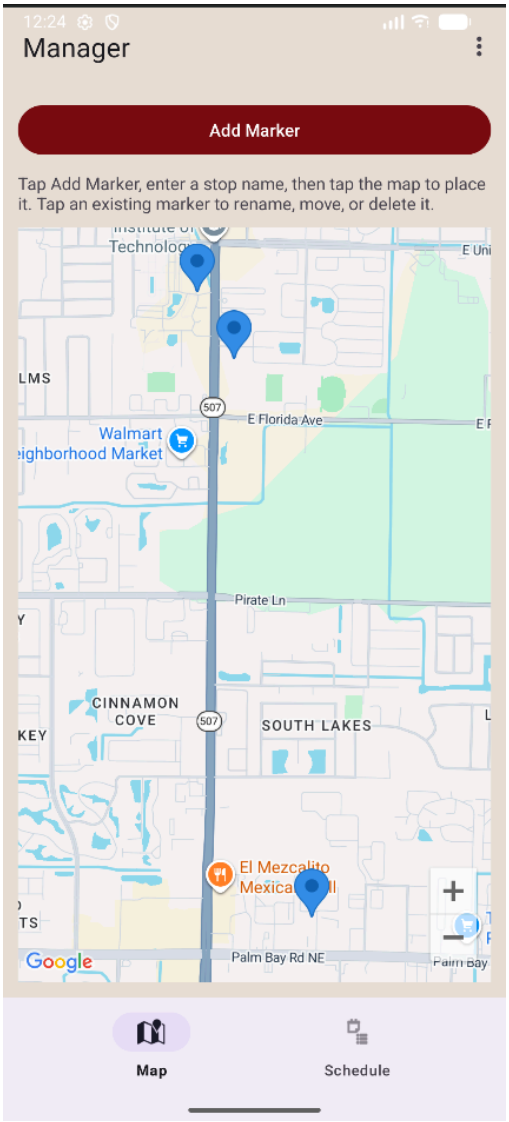
Example workflow: At the start of a shift, the driver opens Settings and enables Share Live Location. The driver then opens the Notify screen and sees that the next stop is PDH Garage at 7:30 PM with an estimated 3 students. The driver selects Panther Bay and 8:15 PM from the dropdown fields and sends the message “Running about 5 minutes late.” Only students whose favorites match that stop and time receive the targeted notification.

3.3 Manager Manual

The manager side is the administrative interface used to control the official shuttle data. It includes a Map tab and a Schedule tab. The manager role is important because the student and driver sides rely on these records for their stop and schedule views.

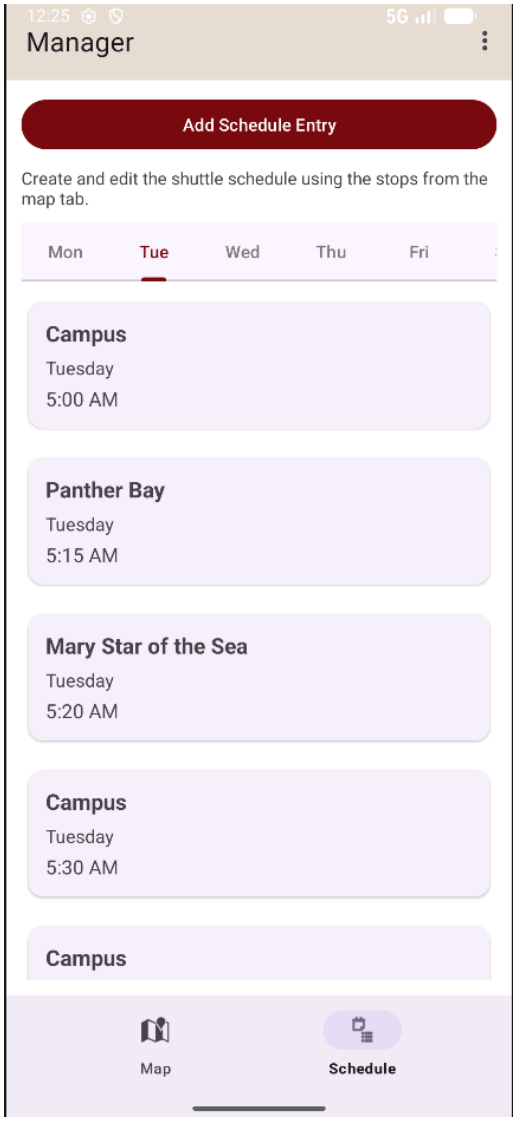
3.3.1 Manager Map Tab

The Manager Map tab allows the manager to add, edit, and delete stop markers. To add a stop, the manager presses the Add Marker button, types a stop name, and taps a location on the map. The new marker is then stored in Firebase. Existing markers can be renamed, moved, or deleted. These changes automatically affect the stop data seen on the student and driver maps.



3.3.2 Manager Schedule Tab

The Manager Schedule tab is used to create and update the official shuttle schedule. The manager can create schedule entries by selecting a stop, choosing a day of the week, and selecting a time. The app also supports selecting multiple days during entry creation, which is useful when the same route pattern repeats across several weekdays. The tab includes weekday filters so the manager can quickly review one day's entries at a time.



3.3.3 Manager Example Workflow

Example workflow: A manager adds a new stop marker for Walmart and then creates Sunday schedule entries for that stop. Because the student and driver schedule screens read from the manager data in Firebase, the new stop and Sunday route times become visible throughout the rest of the app without manual duplication. If the manager later renames the stop, that new name is also reflected in schedule entries tied to the same stop ID.

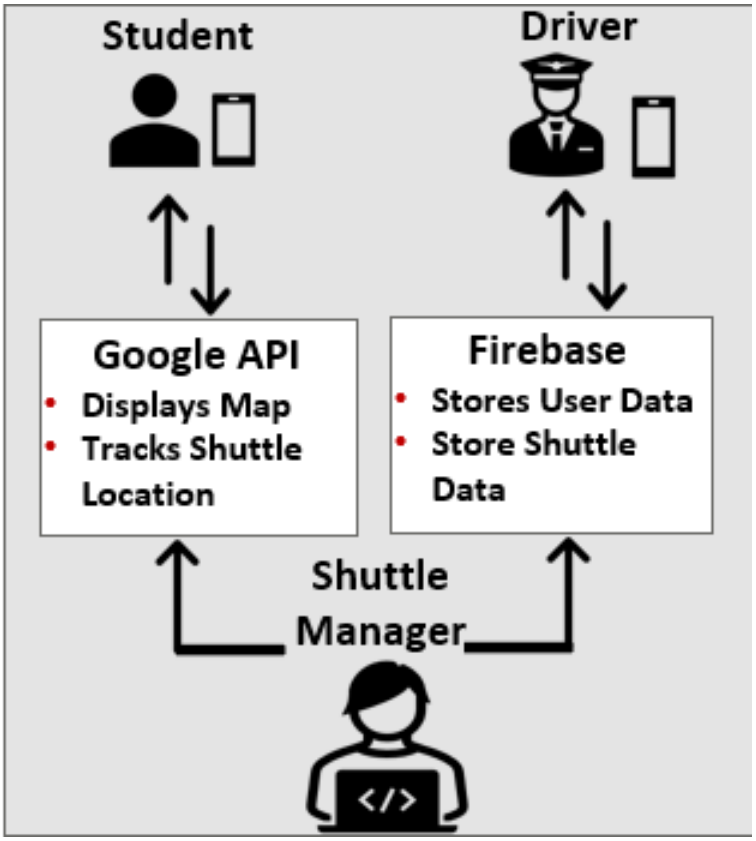
4. Developer Manual

This section documents the internal implementation of the system. It is intended for instructors, future maintainers, and developers who may extend the Panther Shuttle App after the current project is complete.

4.1 System Architecture

The Panther Shuttle App uses a shared mobile-plus-cloud architecture. The Android client is divided into role-based interfaces for students, drivers, and managers. A repository layer handles communication with Firebase. Firebase Authentication is used for anonymous sign-in in the prototype, while Cloud Firestore stores all route-related data. Google Maps is integrated into the app to visualize stops and live location data.

At a high level, managers write official stop and schedule data, drivers publish live location and send notifications, and students consume that shared data through read-only views and personalized filters. Firebase acts as the central synchronization point between all three roles.



4.2 Technologies Used

The application was built in Android Studio using Kotlin as the primary programming language. Material Design components were used to create a consistent Android user interface. Firebase Authentication was used to establish user access, while Cloud Firestore was used as the real-time database. Google Maps API provided the mapping interface for stop markers and live driver location display.

Technology	Purpose
Android Studio	Primary development environment
Kotlin	Main programming language
Firebase Authentication	Prototype sign-in / session handling
Cloud Firestore	Stores stops, schedules, favorites, messages, and live location
Google Maps API	Displays maps, stop markers, and shuttle location
Material Design Components	User interface elements and styling

4.3 Database Design

Cloud Firestore is used as the main backend database. The following collections and documents are central to the application:

stops: Stores manager-created stop markers including stop ID, stop name, coordinates, and update timestamp.

routeSchedule: Stores official shuttle schedule entries including stop ID, stop name, day of week, and time in minutes.

driverMessages: Stores notifications sent by the driver to all students, a stop audience, or a stop-and-time audience.

users/{uid}/favorites: Stores each student's favorite stop and time pairs.

favoriteStopIndex: Stores favorite-stop data in a centralized format to support faster counting and matching.

routes/main/live/driver: Stores the current live driver coordinates and movement metadata.

4.4 Source File Descriptions

The project contains several source files, but the most important files are summarized below. This section should not attempt to describe every minor support file. Instead, it focuses on the files that define role behavior, navigation, Firebase data access, and route management.

File	Description
RoleSelectActivity.kt	Role-selection entry point that routes the user into the Student, Driver, or Manager side.
MainActivity.kt	Main student-side activity that hosts navigation and starts reminder management.
DriverMainActivity.kt	Driver-side activity that hosts driver navigation.
ManagerMainActivity.kt	Manager-side activity that hosts the Map and Schedule tabs.
FirebaseRepo.kt	Primary repository class used to read and write Firebase data across the app.
HomeFragment.kt	Student dashboard showing next stop, ETA, estimated students, and latest driver notifications.
MapsFragment.kt	Student map showing manager-created stop markers and live driver location.
ScheduleFragment.kt	Student schedule page that reads the official manager schedule from Firebase.
NotifyFragment.kt	Student notifications page that displays driver messages relevant to the user.
SettingsFragment.kt	Student settings page for favorite stops and reminder preferences.
DriverMapFragment.kt	Driver map page used for location broadcasting and stop display.
DriverNotifyFragment.kt	Driver notification page with stop/time targeting and next-stop demand estimate.
DriverSettingsFragment.kt	Driver settings page, currently used for Share Live Location.
ManagerMapFragment.kt	Manager map page for adding, moving, renaming, and deleting stops.
ManagerScheduleFragment.kt	Manager schedule page for creating and editing official route times by weekday.

4.5 Important Methods and Functions

Several repository methods are responsible for the main logic of the application. These methods form the connection between the user interface and Firestore.

Method	Purpose
ensureSignedIn()	Ensures Firebase authentication is available before performing database operations.
listenManagerStops()	Reads manager-defined stop markers from Firestore.
createManagerStop()	Creates a new stop marker with name and coordinates.
updateManagerStop()	Updates a stop marker and propagates changes to linked schedule entries.
deleteManagerStop()	Deletes a stop marker and related schedule entries.
listenManagerSchedule()	Reads official schedule entries from Firestore.
createManagerScheduleEntry()	Creates a new schedule entry.
updateManagerScheduleEntry()	Updates an existing schedule entry.
deleteManagerScheduleEntry()	Deletes a schedule entry.
setLiveDriverLocation()	Publishes driver location to the live route document.
listenLiveDriverLocation()	Reads the driver's live location for map display.
sendDriverNotification()	Stores a new driver message with the intended audience.
listenDriverMessagesForAudience()	Reads the messages visible to a student or other role based on audience tags.
countIndexedFavoriteUsersForStopAroundTime()	Estimates demand for a stop using favorite-stop data near a scheduled time.
makeStopTimeAudienceTag()	Creates the audience tag format for stop-and-time-specific notifications.

4.6 Build and Deployment

To build the project, open the application in Android Studio, allow Gradle to sync, and verify that Firebase configuration files and the Google Maps key are present. The app can be run on a physical Android phone or an emulator. Internet access is required for Firebase reads and writes as well as Google Maps functionality.

1. Open the project in Android Studio.
2. Allow Gradle Sync to complete.
3. Verify that Firebase and Maps configuration files are present.
4. Connect an emulator or Android device.
5. Press Run and select the target device.
6. Test all three roles to verify route data sync and navigation.

4.7 Testing and Troubleshooting

Testing focused on both correctness and synchronization. The following behaviors should be verified during project review:

- Role selection routes the user to the correct interface.
- Manager stop updates appear on both the student and driver maps.
- Manager schedule updates appear on both the student and driver schedule-dependent screens.
- Driver live location appears on the student map only when sharing is enabled.
- Student favorite stops can be created from official manager schedule entries.
- Estimated student counts update based on favorite-stop data.
- Driver notifications are filtered correctly for all-users, stop-only, and stop-and-time targeting.
- Dismissed notifications are removed from both the Notifications screen and the Home page summary.

Common troubleshooting points include Firebase permission errors, missing or restricted Maps API keys, notification permission issues on Android 13 or newer, and emulator location settings. Another common issue during development is stale data caused by not resyncing after Firebase rule or schema changes. In those cases, rebuilding the app, reopening the correct role screen, or refreshing the test data in Firebase often resolves the problem.

5. Future Work

Several future improvements could strengthen the system. First, reminder notifications could be made more reliable in the background by integrating WorkManager or AlarmManager so reminders still fire even if the app is fully closed. Second, stronger role-based access control could be added so only approved drivers and managers can access sensitive features such as live location sharing or schedule editing. Third, analytics could be added to the manager side to identify high-demand stops and route trends. Additional features could include support for multiple shuttle routes, service-delay prediction based on live movement, and accessibility improvements such as larger-text display modes or voice-based announcements.

6. Conclusion

The Panther Shuttle App demonstrates a practical mobile solution for campus transportation management. By combining separate Student, Driver, and Manager experiences with a shared Firebase backend, the application provides real-time communication, centralized route control, and personalized stop information. The student role improves visibility and convenience, the driver role supports communication and route operation, and the manager role provides administrative control over stops and schedules. Together, these components form a unified system that can improve campus shuttle usability and encourage broader adoption of campus transportation.

This document has described both how to use the application and how the application is built. It can serve as a submission-ready manual, a guide for demonstrations, and a starting point for future teams or developers who may continue extending the project.

Appendix A. Role-Based Usage Examples

This appendix provides additional role-based examples that can be used during a live demonstration or oral presentation. These examples are useful because they show how the three sides of the application interact with one another through shared route data. Instructors and evaluators often want to see not only individual screens, but also how a change in one role affects the rest of the system.

A.1 Student Example Set

Scenario 1: A student wants to decide when to leave a dorm room. The student opens the Home screen and sees that the next stop is PDH Garage in 8 minutes. The student then opens the Map screen and confirms that the live driver marker is still a short distance away. Based on this information, the student knows there is enough time to walk to the stop.

Scenario 2: A student receives a targeted message from the driver. The student has Panther Bay at 8:15 PM saved as a favorite stop. The driver sends a notification specifically to Panther Bay at 8:15 PM, warning that the shuttle is running five minutes late. The student receives this message because the favorite-stop tag matches the notification audience.

Scenario 3: A student wants an early reminder before a stop. The student enables the 'Notify Before Favorite Stops' setting. When an upcoming manager-defined schedule entry matches a saved favorite stop and time, the app generates a reminder approximately 15 minutes before the stop. This helps reduce missed rides.

A.2 Driver Example Set

Scenario 1: A driver begins a shift. The driver opens the Settings screen and enables Share Live Location. The Driver Map screen confirms that live broadcasting is active. Students can now see the driver marker on the student map.

Scenario 2: A driver communicates with a selected group of riders. The Driver Notify screen displays the next official stop and estimated student count. The driver selects a specific stop and time from the dropdowns and sends a message only to students who have that matching favorite-stop profile. This prevents unrelated riders from receiving unnecessary alerts.

Scenario 3: A driver stops sharing location at the end of a shift. The driver turns off Share Live Location in the settings page. The system stops publishing coordinates to Firebase and clears the live route marker so students do not see stale location data after service has ended.

A.3 Manager Example Set

Scenario 1: A manager needs to add a temporary weekend stop. The manager opens the Manager Map tab, creates a new stop marker, and then opens the Manager Schedule tab to add Sunday schedule entries for that stop. Once saved, the stop becomes visible on student and driver maps and the new schedule entry appears on the student schedule page.

Scenario 2: A manager corrects a stop name. The manager renames a stop marker on the map. Because schedule entries are associated with the stop ID and stop name, the update is propagated to linked schedule entries so that the corrected label appears across the app.