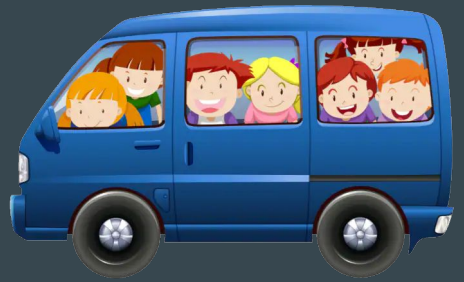
Panther Carpooling & Parking

Milestone 1



Team Members

- Austin Phillips — aphillips2022@my.fit.edu
- Hunter Smith — jsmith2022@my.fit.edu
- Jacqueline Torres — jtorres2020@my.fit.edu
- Jason Smith — hsmith2021@my.fit.edu

Faculty Advisor and Client

- Dr. Philip Chan — pkc@cs.fit.edu

Overview of Milestone 1

- Select technical tools for database, front end, and back end
- Select technical tools for code version and control testing
- Resolve technical challenges
- Create documents
  - Requirement
  - Design
  - Test Plan
- Overall Progress of Milestone 1
- Expectation of Milestone 2

Technical Tools

Database, Front End, and Back End

- Database
  - SQLite
- Front and Back End
  - Blazor Server Application
  - C#

Code Version Control and Testing

- Integration testing
  - Azure and Github Integration
- Unit testing
  - bUnit
- Performance testing
  - Visual Studio Builds
- Version Control
  - Github

Technical Challenges

Resolved Challenges:

Challenge: Accessing Google Maps/Routing API
Challenge: Accessing calendar API

Unresolved Challenges:

Challenge: Accessing Florida Tech tracks account API
Resolution: Email verification of school email

Requirements

Functional Requirements

1. Expansive User Profile
    - Basic information
    - Location information
    - Carpooling Schedule
    - Vehicle information (Drivers only)
    - Carpooling Preferences
    - Friends List
2. Creating Group Recommendations
    - Considers User information
    - Driver automatically suggested
3. Group Formation
    - Using system recommendations
    - Manually form group
4. Trip Lifecycle
    - Before, during and after trip

Interface Requirements

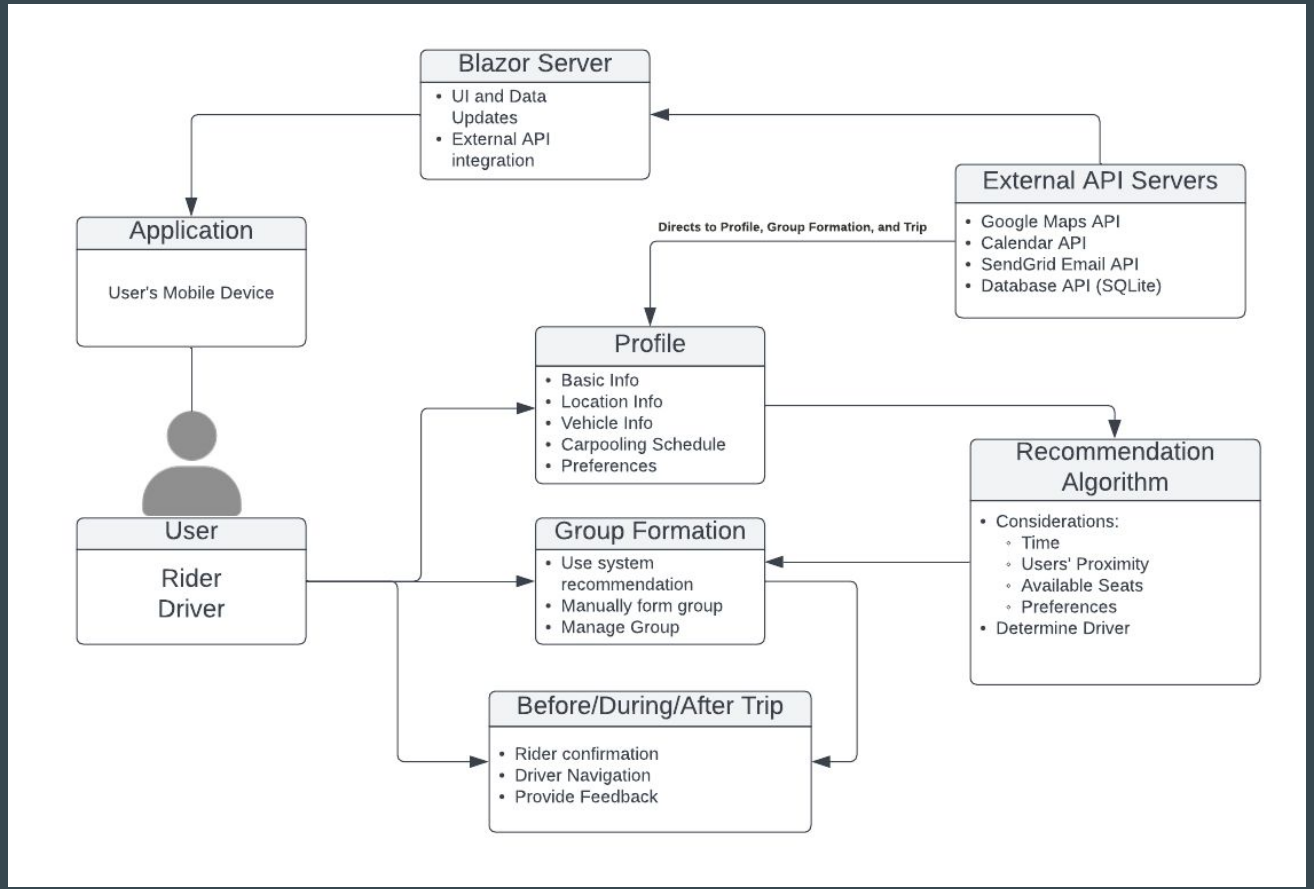
- Functional on web and mobile
- Light and Dark mode
- 2 Interfaces for the User
  - Rider
  - Driver
- System Administrator View
  - Groups, Users, Available preferences

Security Requirements

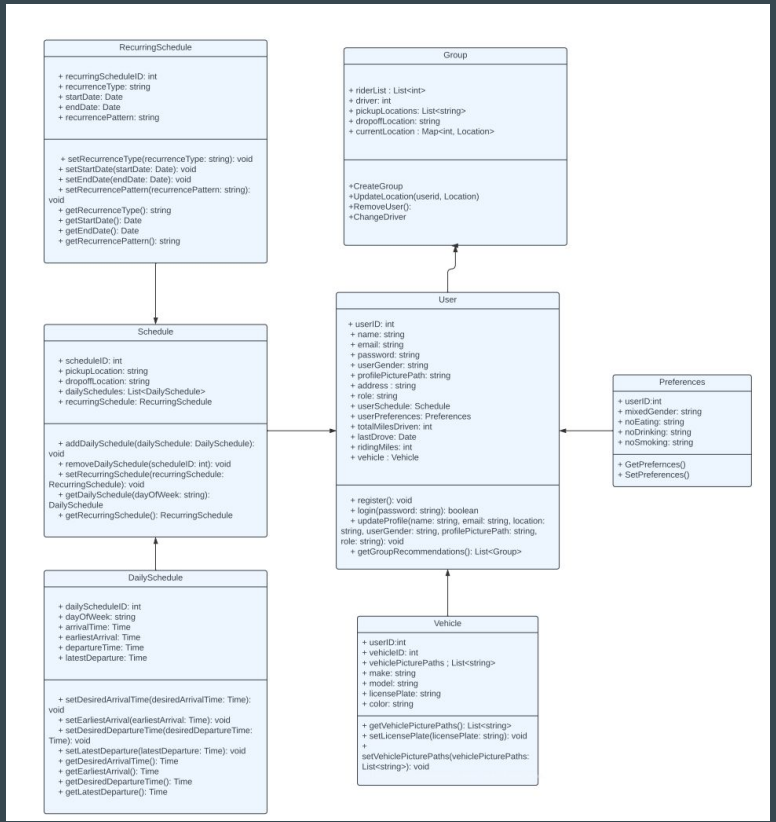
- Florida Tech Email Address Required
- System Admin Access
  - Database information
  - Reports
- Block Users
- Rider Pick-Up Confirmation
- Linking to Florida Tech Safe app
  - Contact Local Authorities
  - Share Location with Authorities

Design

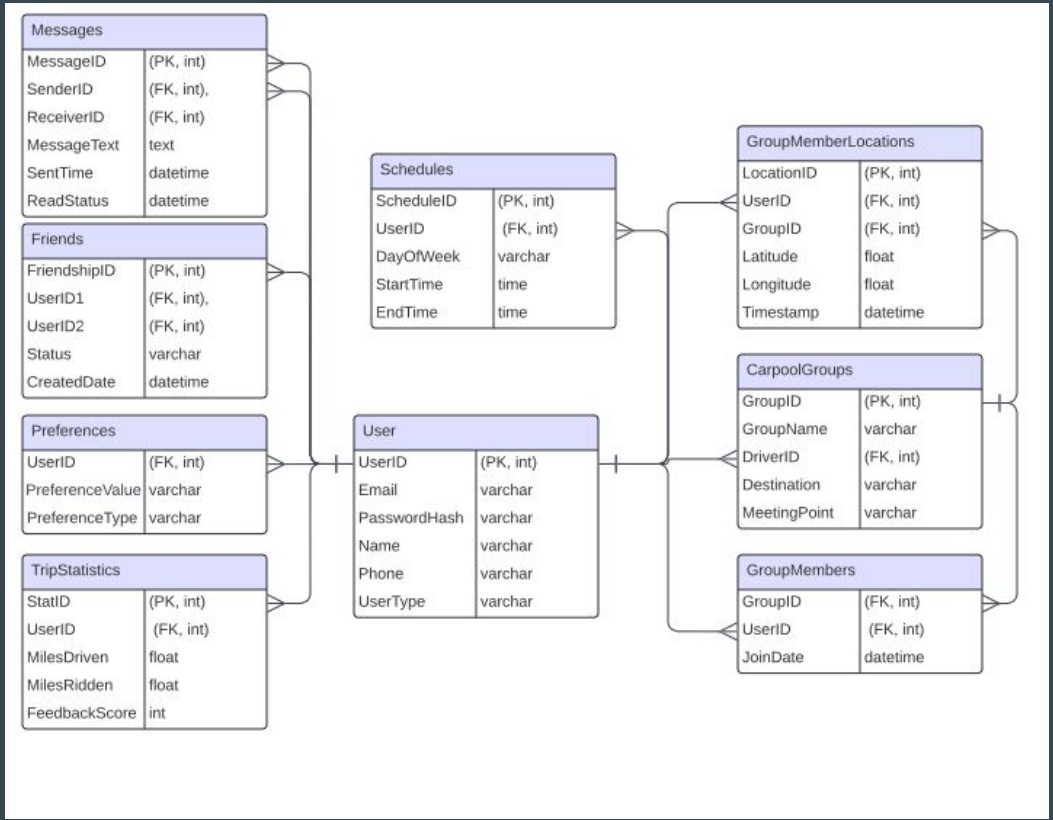
System Architecture Diagram



UML Diagram


RecurringSchedule
+ recurringScheduleID: int
+ recurrenceType: string
+ startDate: Date
+ endDate: Date
+ recurrencePattern: string
+ setRecurrenceType(recurrenceType: string): void
+ setStartDate(startDate: Date): void
+ setEndDate(endDate: Date): void
+ setRecurrencePattern(recurrencePattern: string): void
+ getRecurrenceType(): string
+ getStartDate(): Date
+ getEndDate(): Date
+ getRecurrencePattern(): string
Group
+ riderList : List<int>
+ driver: int
+ pickupLocations: List<string>
+ dropoffLocation: string
+ currentLocation : Map<int, Location>
+CreateGroup
+UpdateLocation(userid, Location)
+RemoveUser():
+ChangeDriver
Schedule
+ scheduleID: int
+ pickupLocation: string
+ dropoffLocation: string
+ dailySchedules: List<DailySchedule>
+ recurringSchedule: RecurringSchedule
+ addDailySchedule(dailySchedule: DailySchedule): void
+ removeDailySchedule(scheduleID: int): void
+ setRecurringSchedule(recurringSchedule: RecurringSchedule): void
+ getDailySchedule(dayOfWeek: string): DailySchedule
+ getRecurringSchedule(): RecurringSchedule
User
+ userID: int
+ name: string
+ email: string
+ password: string
+ userGender: string
+ profilePicturePath: string
+ address : string
+ role: string
+ userSchedule: Schedule
+ userPreferences: Preferences
+ totalMilesDriven: int
+ lastDrove: Date
+ ridingMiles: int
+ vehicle : Vehicle
+ register(): void
+ login(password: string): boolean
+ updateProfile(name: string, email: string, location: string, userGender: string, profilePicturePath: string, role: string): void
+ getGroupRecommendations(): List<Group>
Preferences
+ userID:int
+ mixedGender: string
+ noEating: string
+ noDrinking: string
+ noSmoking: string
+ GetPrefernces()
+ SetPreferences()
DailySchedule
+ dailyScheduleID: int
+ dayOfWeek: string
+ arrivalTime: Time
+ earliestArrival: Time
+ departureTime: Time
+ latestDeparture: Time
+ setDesiredArrivalTime(desiredArrivalTime: Time): void
+ setEarliestArrival(earliestArrival: Time): void
+ setDesiredDepartureTime(desiredDepartureTime: Time): void
+ setLatestDeparture(latestDeparture: Time): void
+ getDesiredArrivalTime(): Time
+ getEarliestArrival(): Time
+ getDesiredDepartureTime(): Time
+ getLatestDeparture(): Time
Vehicle
+ userID:int
+ vehicleID: int
+ vehiclePicturePaths ; List<string>
+ make: string
+ model: string
+ licensePlate: string
+ color: string
+ getVehiclePicturePaths(): List<string>
+ setLicensePlate(licensePlate: string): void
+ setVehiclePicturePaths(vehiclePicturePaths: List<string>): void

Entity Relationship Diagram



Test Plan

Functional Tests

We've come up with a rigorous testing plan, the make sure every functional requirement works with many different scenarios, like...

- Correct Inputs
- Incorrect Inputs
- Rider vs Driver
- Security Settings

We're putting extra focus on security, to make sure our users are safe.

User Tests

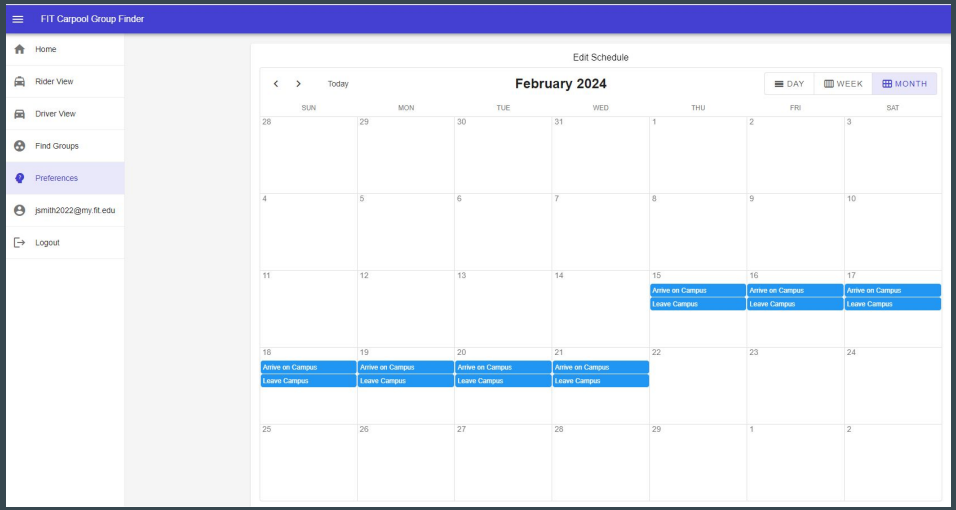
In addition, we've come up with some user tests.
This will ensure that our app is usable and intuitive to the average user.

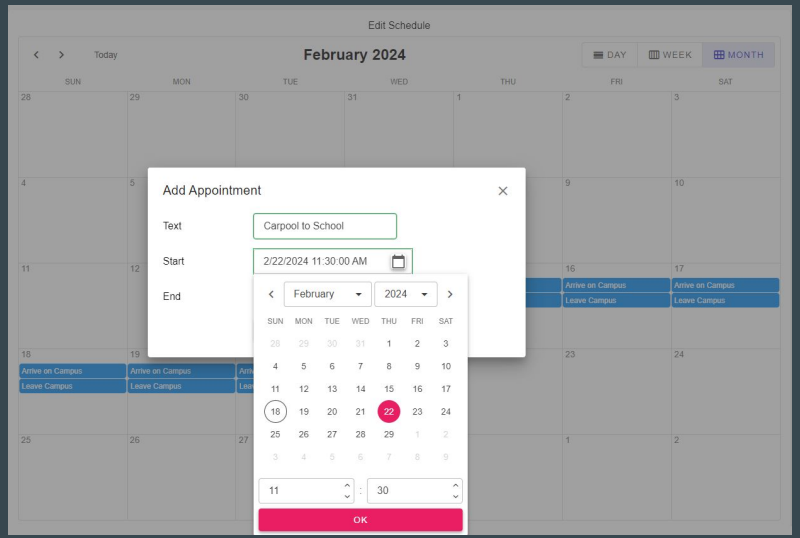
Example: Matching Algorithm

We will ask users to fill out their preferences, then rank a list of fake groups from best to worst. We can then adapt our matching algorithm to their choices as closely as possible.

Demos

Calendar API



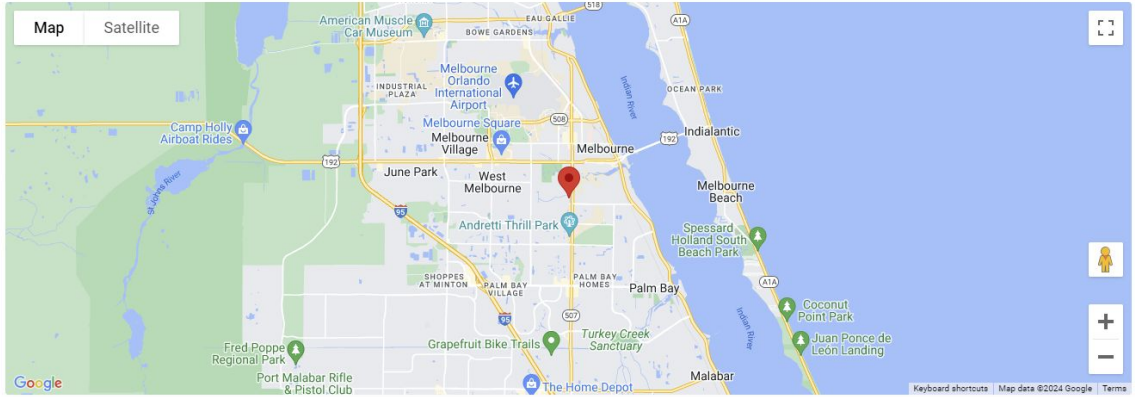


Google Maps API - Rider Page

FIT Carpool Group Finder

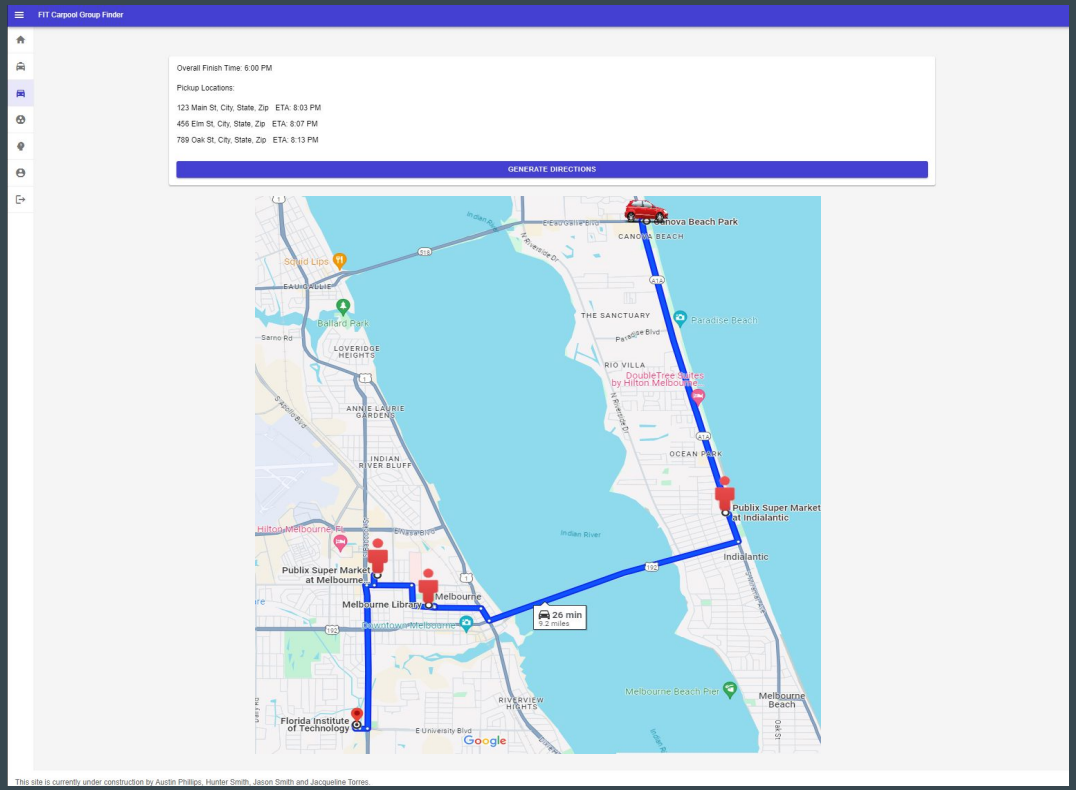
Driver Location:

Melbourne, FL



This site is currently under construction by Austin Phillips, Hunter Smith, Jason Smith and Jacqueline Torres.

Google Maps API - Driver Page



Progress Matrix of Milestone 1

Task	Completion	Austin	Jason	Jacqueline	Hunter	To do:
Technical tools- database, front end, back end	100%	25%	25%	25%	25%	N/A
Technical tools - code version control, testing	100%	25%	25%	25%	25%	N/A
Technical challenges	75%	55%	15%	15%	15%	Implement Florida Tech email account verification
Requirement Document	90%	15%	55%	15%	15%	Implement Advisor Feedback
Design Document	90%	15%	15%	55%	15%	Implement Advisor Feedback
Test Plan Document	95%	15%	15%	15%	55%	Update for new requirements

Milestone 2

- Develop and test user interface for first half of user profile setup (input schedule, location, other preferences)
- Develop and test second half of user profile (friendlist, privacy, user statistics...)
- Develop and test database integration into the web application
- Develop and test routing/map integration using assumed groups

Thoughts, Questions, Suggestions?

View Site