Introduction:

WanderSync: A Collaborative Travel Management System makes traveling simple. Whether for solo or group travel, a quick business trip, a weekend holiday, or well deserved month long vacation, it's the app you want to choose to manage your travel. WanderSync takes the hassle of juggling transportation options, dining reservations, travel locations with everyone in your party by organizing and updating everything you need into one app. Our software is user-friendly and promotes real-time collaboration to keep everyone in sync.

This project is our Fall 2024 CS 2340 group project at Georgia Tech. WanderSync is a fully functional android app, developed using Android Studio, programmed in Java, and hosted on Github. We used Google Firebase to implement real-time collaboration. We learned about the software development process, specifically Agile development. We created UML diagrams, practiced iterating our project in sprints, wrote test cases for our program, and presented our work in both writing and in live in-person demos.

Design & Architecture:

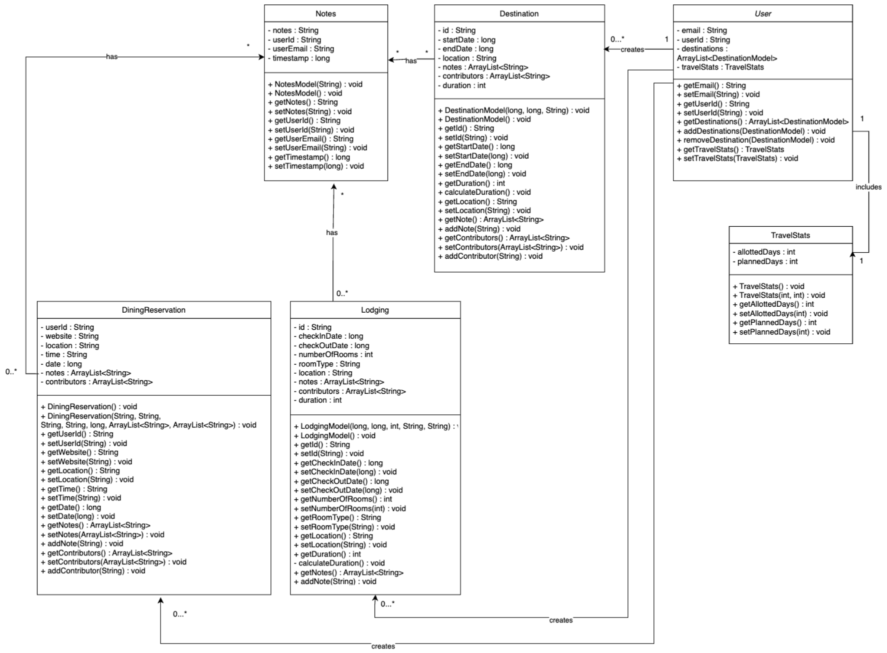
Aiding WanderSyncs development was our commitment to prioritizing project definition and design to establish an effective workflow. Keeping track of our app's objectives and progress through a collaborative environment such as Notion, then representing functionality, user interaction and use cases through UML diagrams before implementation lead to a smooth development process.

To give more insight in our pre-implementation design and more of WanderSyncs functionality, we showcase two diagrams.

Design Class Diagram

A design class diagram represents the relationships between classes, their methods and attributes to the project as a whole. Our DCD (used in our third sprint) shows that a User includes travel statistics. They can create destinations, dining reservations and room accommodations and add notes for each of these.

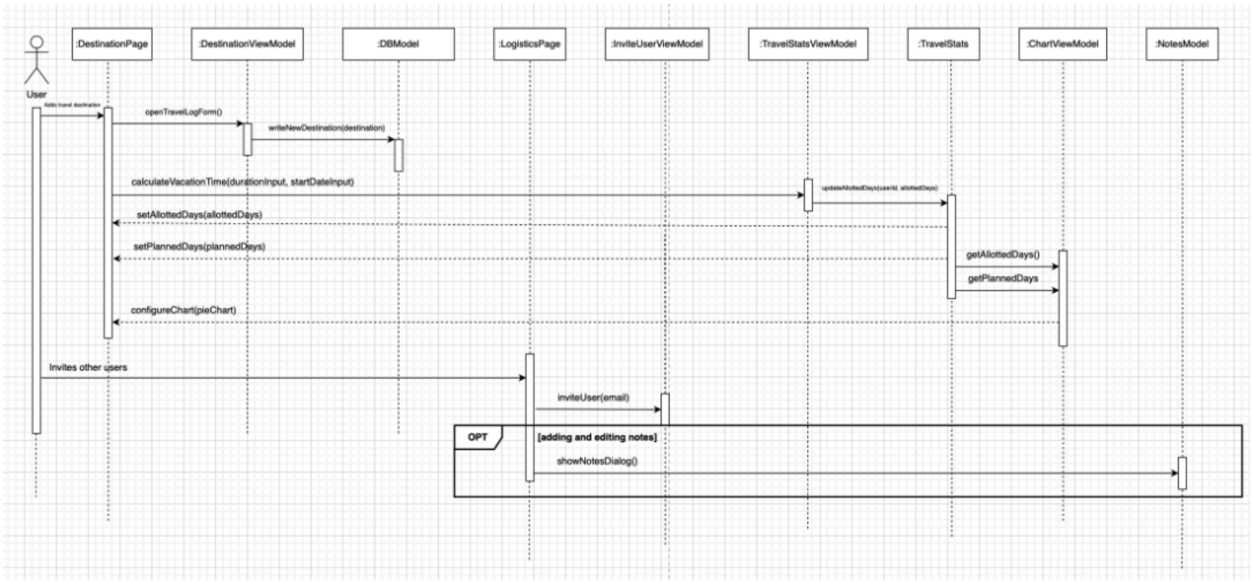
Let's highlight some of the methods and attributes of the class DiningReservation. A dining reservation has a website, location, date and time. A user is able to create a dining reservation, input and view information about the reservation, and add notes and other participants.


Notes
- notes : String
- userId : String
- userEmail : String
- timestamp : long
+ NotesModel(String) : void
+ NotesModel() : void
+ getNotes() : String
+ setNotes(String) : void
+ getUserId() : String
+ setUserId(String) : void
+ getUserEmail() : String
+ setUserEmail(String) : void
+ getTimestamp() : long
+ setTimestamp(long) : void
Destination
- id : String
- startDate : long
- endDate : long
- location : String
- notes : ArrayList<String>
- contributors : ArrayList<String>
- duration : int
+ DestinationModel(long, long, String) : void
+ DestinationModel() : void
+ getId() : String
+ setId(String) : void
+ getStartDate() : long
+ setStartDate(long) : void
+ getEndDate() : long
+ setEndDate(long) : void
+ getDuration() : int
+ calculateDuration() : void
+ getLocation() : String
+ setLocation(String) : void
+ getNote() : ArrayList<String>
+ addNote(String) : void
+ getContributors() : ArrayList<String>
+ setContributors(ArrayList<String>) : void
+ addContributor(String) : void
User
- email : String
- userId : String
- destinations :
ArrayList<DestinationModel>
- travelStats : TravelStats
+ getEmail() : String
+ setEmail(String) : void
+ getUserId() : String
+ setUserId(String) : void
+ getDestinations() : ArrayList<DestinationModel>
+ addDestinations(DestinationModel) : void
+ removeDestination(DestinationModel) : void
+ getTravelStats() : TravelStats
+ setTravelStats(TravelStats) : void
TravelStats
- allottedDays : int
- plannedDays : int
+ TravelStats() : void
+ TravelStats(int, int) : void
+ getAllottedDays() : int
+ setAllottedDays(int) : void
+ getPlannedDays() : int
+ setPlannedDays(int) : void
DiningReservation
- userId : String
- website : String
- location : String
- time : String
- date : long
- notes : ArrayList<String>
- contributors : ArrayList<String>
+ DiningReservation() : void
+ DiningReservation(String, String,
String, String, long, ArrayList<String>, ArrayList<String>) : void
+ getUserId() : String
+ setUserId(String) : void
+ getWebsite() : String
+ setWebsite(String) : void
+ getLocation() : String
+ setLocation(String) : void
+ getTime() : String
+ setTime(String) : void
+ getDate() : long
+ setDate(long) : void
+ getNotes() : ArrayList<String>
+ setNotes(ArrayList<String>) : void
+ addNote(String) : void
+ getContributors() : ArrayList<String>
+ setContributors(ArrayList<String>) : void
+ addContributor(String) : void
Lodging
- id : String
- checkInDate : long
- checkOutDate : long
- numberOfRooms : int
- roomType : String
- location : String
- notes : ArrayList<String>
- contributors : ArrayList<String>
- duration : int
+ LodgingModel(long, long, int, String, String) :
+ LodgingModel() : void
+ getId() : String
+ setId(String) : void
+ getCheckInDate() : long
+ setCheckInDate(long) : void
+ getCheckOutDate() : long
+ setCheckOutDate(long) : void
+ getNumberOfRooms() : int
+ setNumberOfRooms(int) : void
+ getRoomType() : String
+ setRoomType(String) : void
+ getLocation() : String
+ setLocation(String) : void
+ getDuration() : int
- calculateDuration() : void
+ getNotes() : ArrayList<String>
+ addNote(String) : void
has
*
0..*
*
has
*
0...*
creates
1
1
includes
1
*
has
0..*
0...*
creates
0...*
creates

Sequence Diagram

A sequence diagram is a way to represent a use case broken down into methods and a clear flow. This SD shows the fundamental use case of WanderSync – a user creating and modifying a destination.

It starts by the user adding a new travel destination. This triggers a new form to input the destination information. Once created, our app takes the data in the form and displays the information, including calculated data, such as the total vacation duration on the Travel Logs page. We also store this data and create visualizations in our travel statistics section. Finally, our SD shows a user inviting collaborators via email. Optionally, a user can add and edit notes for the destination.



User Interface (UI):

[@ Tina : present a visual tour of the app through screenshots showcasing various screens focusing on the main functionalities and interaction flow within the app]

Functionality:

[@ Tina : link to a video demo of app's functionality with insight to features and UI]

Conclusions and Reflections/ Learning:

We are very proud of WanderSync and what our team has accomplished in a short semester. Not only did we consistently complete the rubric requirements for each sprint, we went above and beyond to make WanderSync a simple and clean app — a piece of software our own team members would be interested in downloading.

Our group prioritized collaboration, we found it non-negotiable that each team be involved in every step of the process, from the requirements engineering to implementation. We utilized our group members strengths and everyone excelled in their own domain. We acknowledge everyone's contributions in each step, however we also feel it is just as important to highlight where our members shined the brightest. Julian and Andres were our foundation. They worked on the core of the software and nailing the more challenging parts of implementation. They also tended to take on a leadership role and consistently helped guide the project. Benjamin and Zachary were the glue of the team. They were key in implementing the finer, but crucial parts of the software that pushed our app to the next level. Not only did they check that the code was readable, and adhered to good program design such as the SOLID and GRASP principles, but they also functionally validated the code through test cases. Joni and Tina were the cornerstones of the group. They were essential in maintaining a smooth development process. They would translate the engineering requirements into UML diagrams and user cases that substantially made implementation more effective. They also focused on turning our project from just software into an actual product, by writing user-friendly explanations of WanderSync's functionality and compiling and documenting the developer process.

Our team was hard-working and goal-oriented from the start to end, however, that doesn't mean that we didn't also run into some challenges. During one of the sprints, our team collaboration was not as high as it needed to be. This led to unnecessary stress and great pressure for those who had work unfairly hoisted upon them. Though our collaboration fell through, our communication had not. Immediately after we had completed our deadlines, we had a group discussion about how we could improve and avoid any more complications in the future. This small hitch in our journey actually led the team to be stronger and more productive than we had any of the previous sprints.

This project was a holistic overview of software development, thus, as a team, we learned immeasurably valuable information that will be utilized throughout our computer science education and career. We got real world experience through being on a team and creating actual software. We applied the industry principles requirements engineering, design, implementation and verification and validation taught in lecture. We used the agile software process and iterated our work by using one to two week long sprints. We hosted our project on github, kept a Notion where we kept track of our backlog and deadlines. We designed UML diagrams and translated them into code, using Android Studio in the language Java to create our app. We required our code be validated by another team member before pushing it into main. This encouraged constant collaboration and solid teamwork throughout the semester.

Contributors:
Everyone was instrumental to both developing the final project and contributing to the showcase website's deployment.

Thank you,
Julian McCorkle, Andres Quast, Benjamin Liu, Zachary Doran, Tina Chen, Jacqueline "Joni" Isbell