GreekRight: Final Product Submission

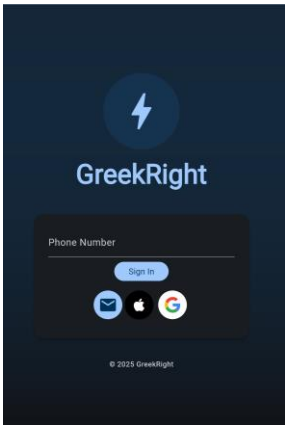
Simon Ulloa and Vanessa Eichensehr

Product Description

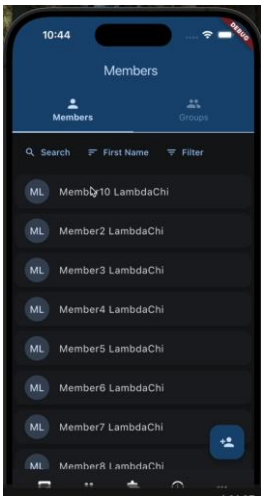
GreekRight is an application for fraternity and sorority management. Our application allows student leaders to manage their organization and allows members to access the information that they need. Our current solution has three major components: a calendar system to keep members informed of events, a shifts system to coordinate volunteers, and a messaging system for streamlined communication. We plan to continue to add features and allow each organization to choose and customize the features that best suit their needs.

Product Functionalities

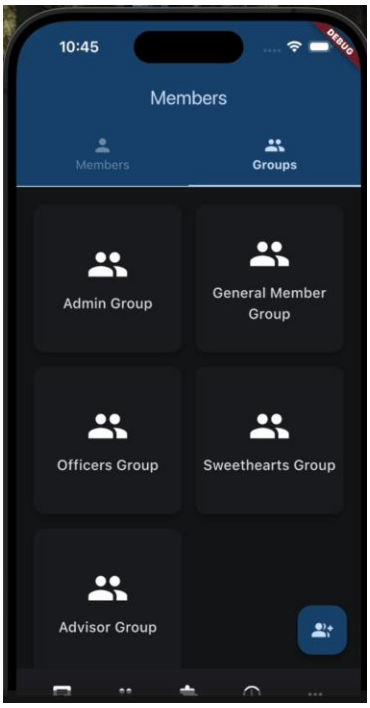
1. Log In



2. View Members

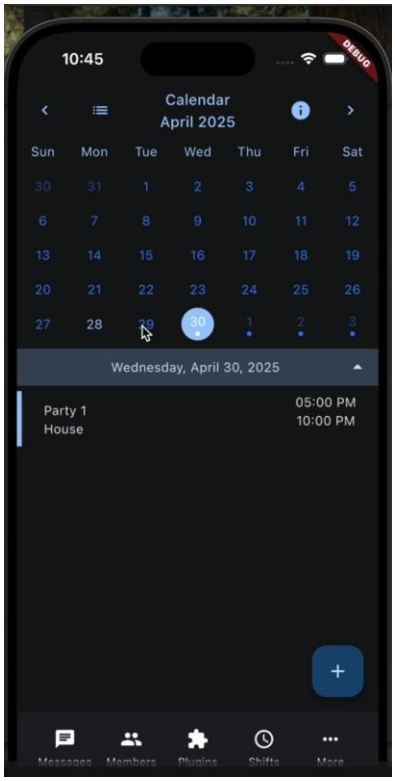


3. Customize Groups

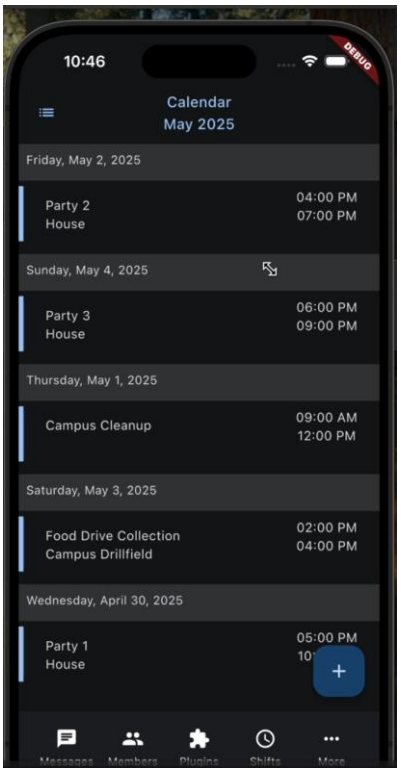


Calendar System

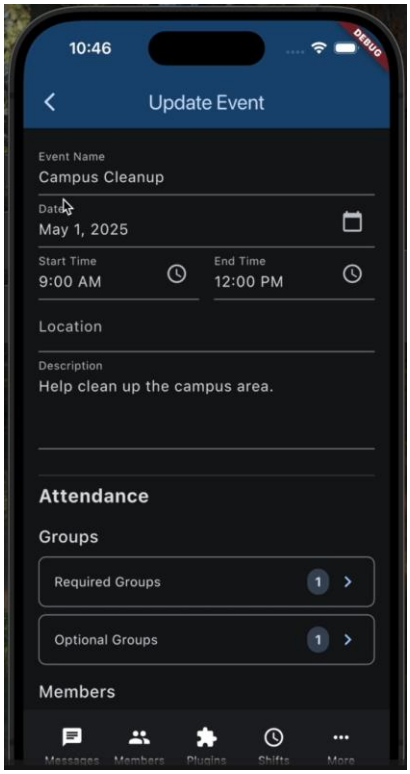
4. Calendar



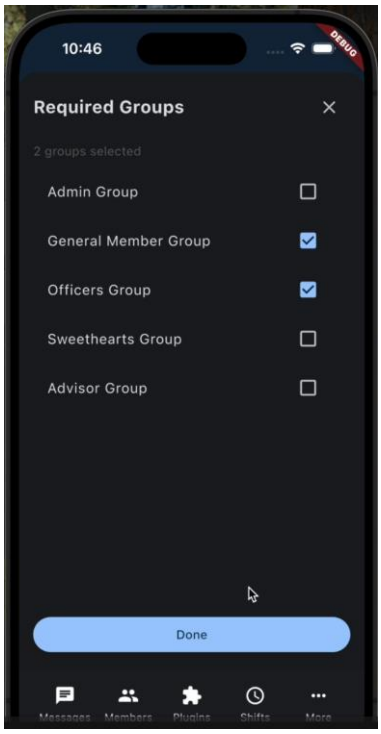
5. Event List



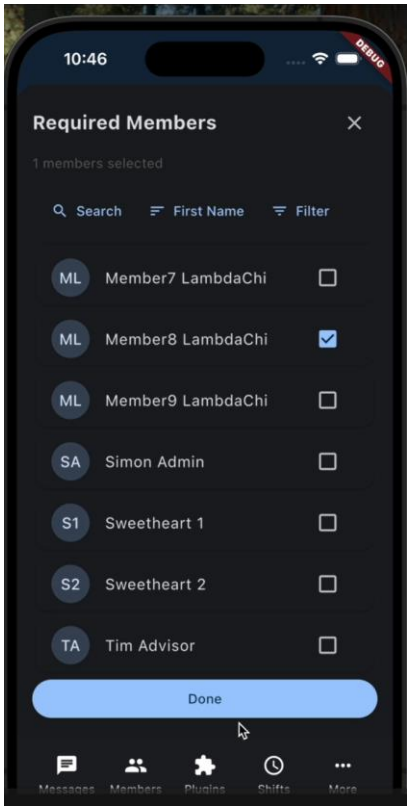
6. Create/Update Event (similar screen, titled differently)



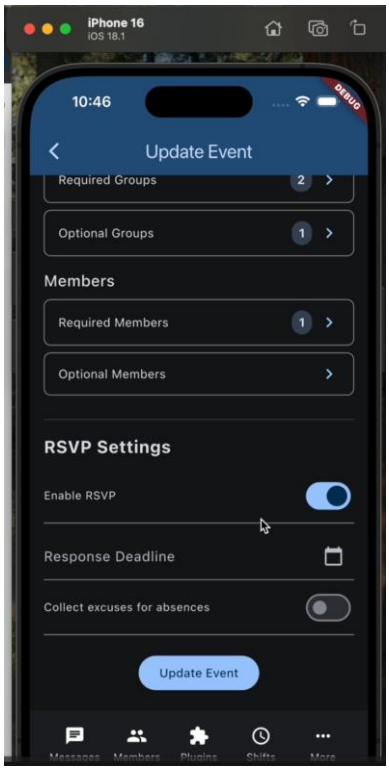
7. Set Required Groups for Attendance at Event



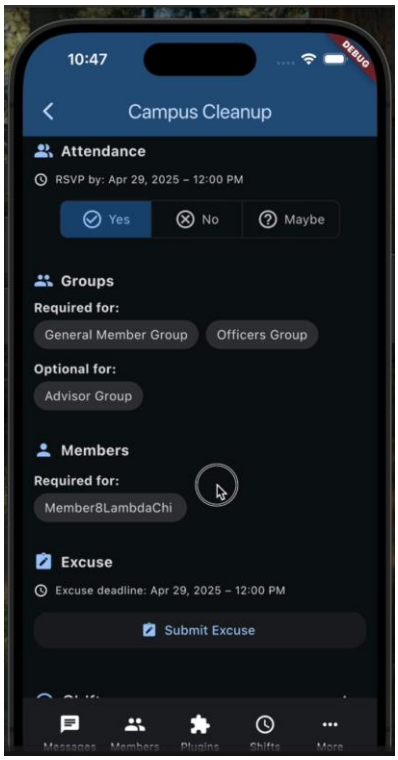
8. Set Required Members for Attendance at Event



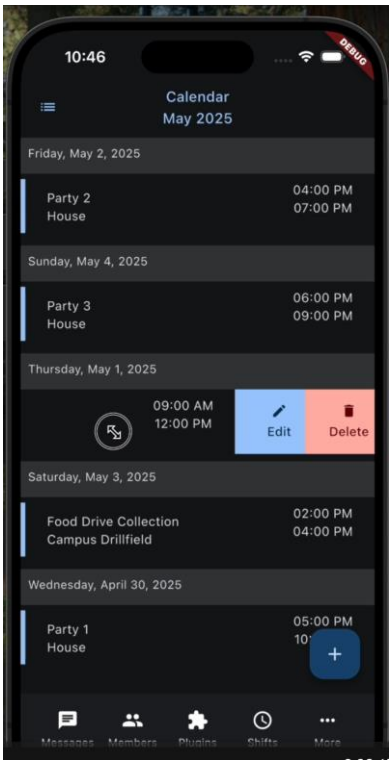
9. Set up RSVPs for Event



10. RSVP for Event

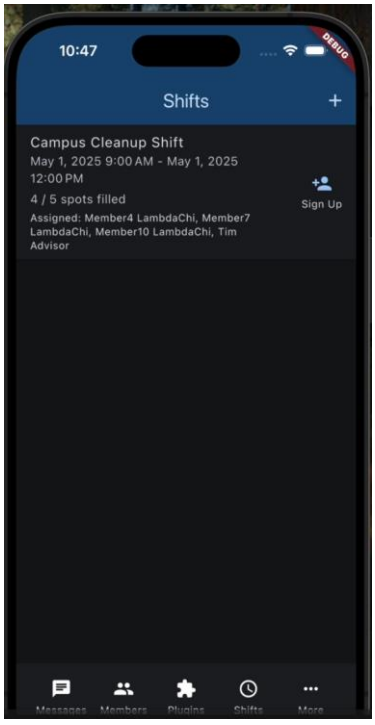


11. Delete Event (Slide to delete on Calendar Screen or Event List View)

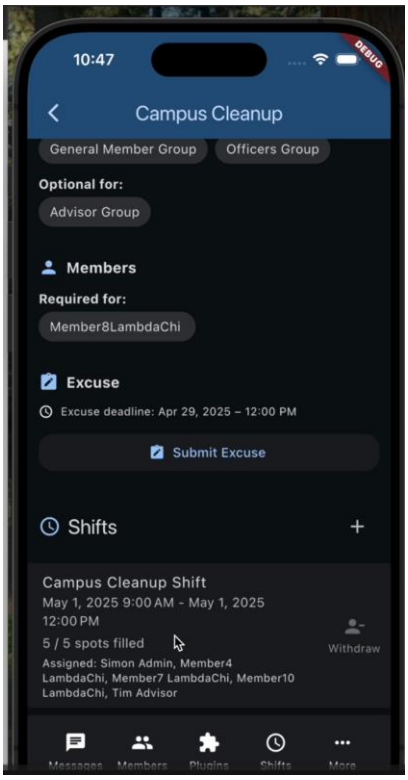


Shifts System

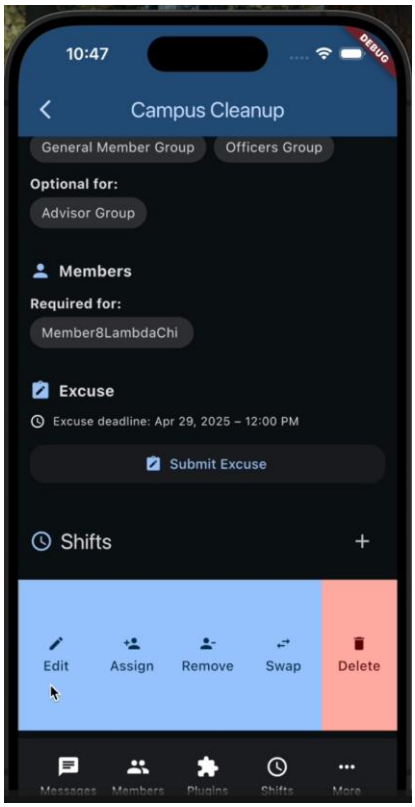
12. Shifts Screen



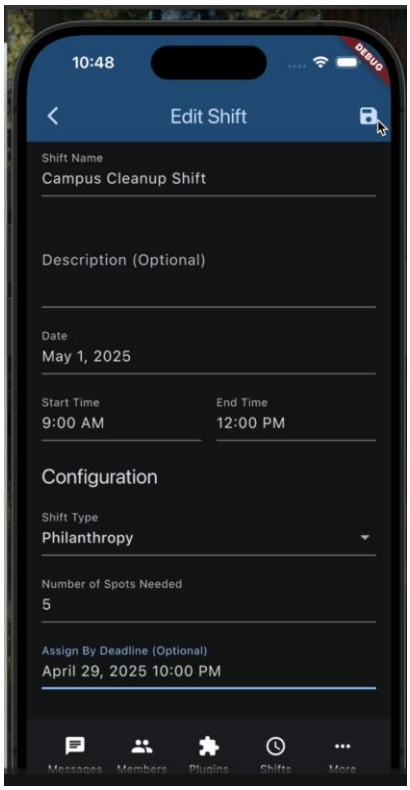
13. Signing Up for a Shift Linked to an Event



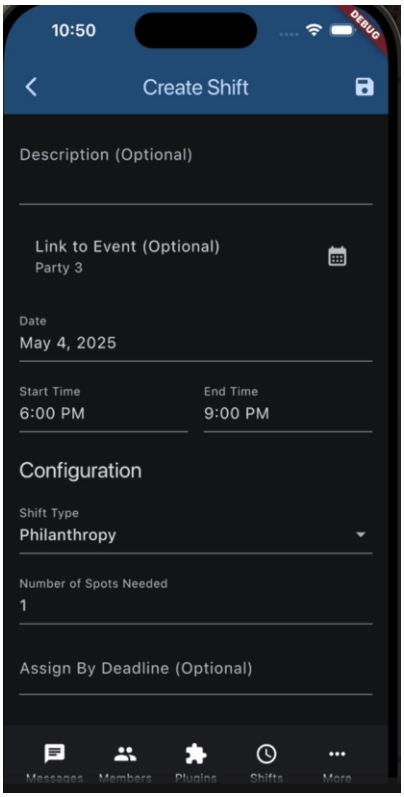
14. Shift Options (Edit, Assign, Remove, Swap, Delete)



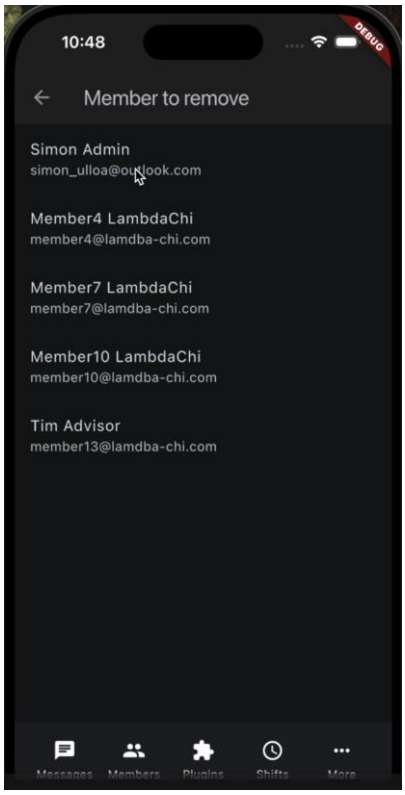
15. Edit/Create Shift that is linked to an event



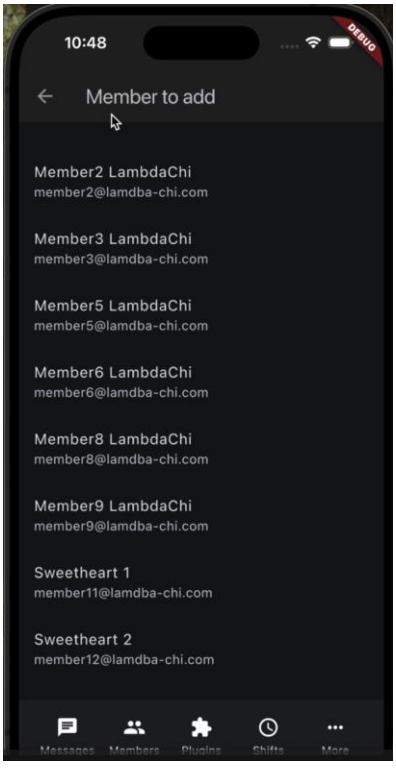
16. Edit/Create Shift not linked to an event



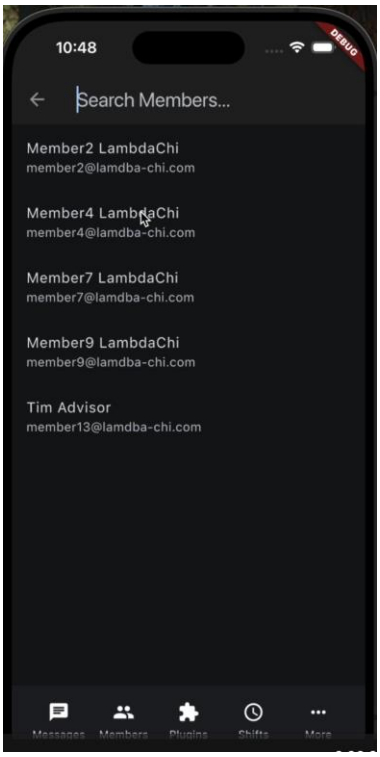
17. Swap Shifts



18. Swap Shifts (second screen)

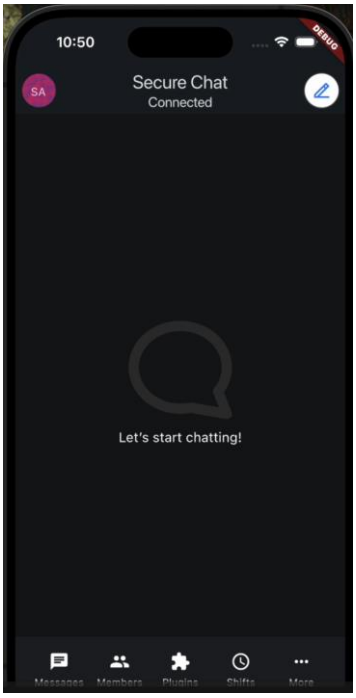


19. Search Members (to assign and remove from shifts)

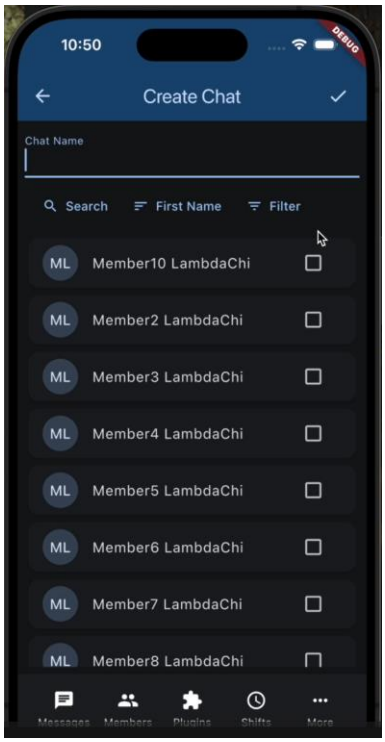


Messages System

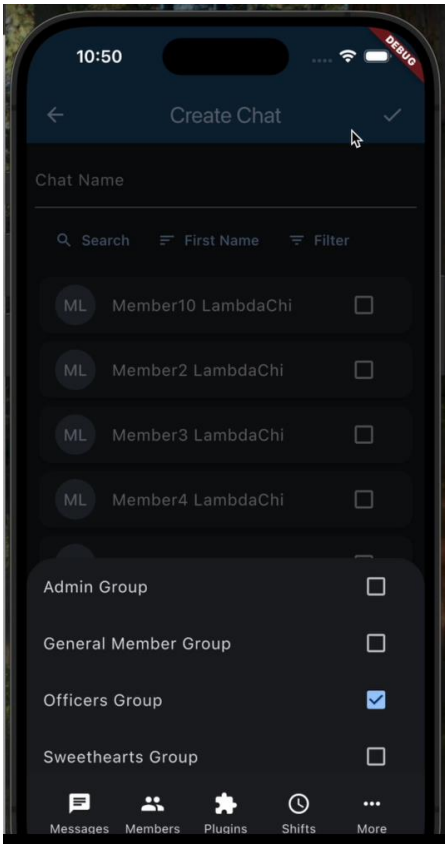
20. Messages Home Screen (with no chats created yet)



21. Create Group Chats (Add Individual Members)

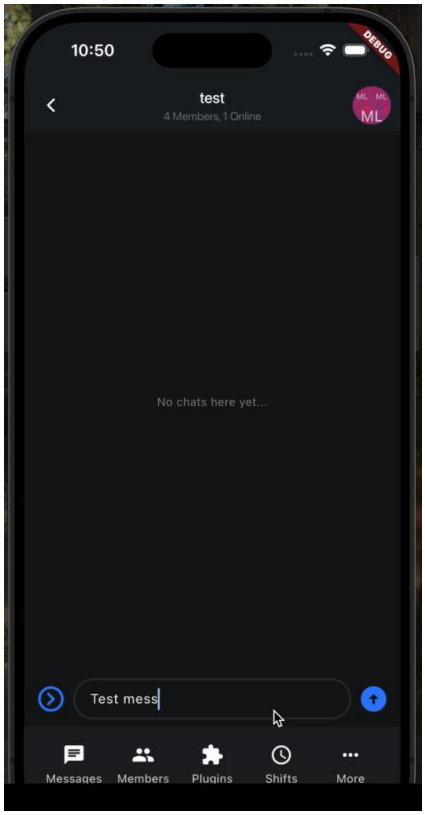


22. Create Group Chats (Add entire group of people)



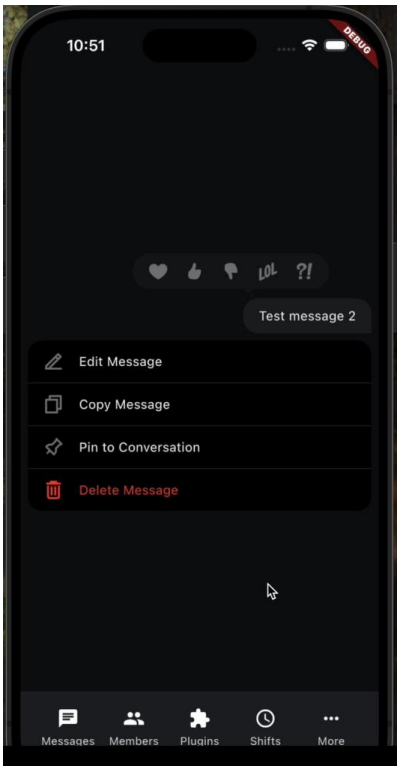
23. Click on a chat

24. Type a Message

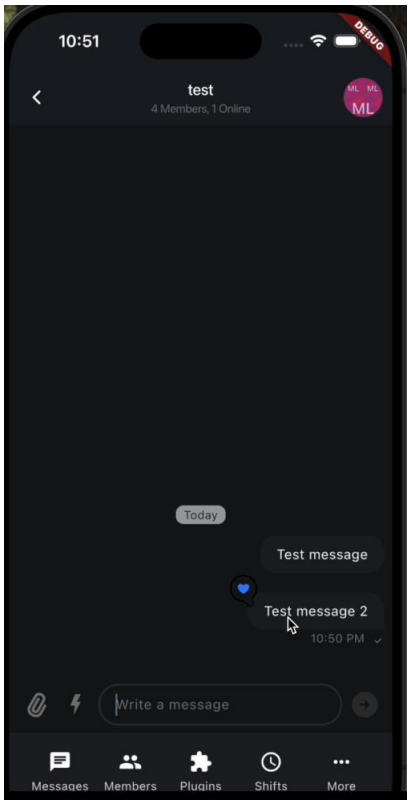


25. Send Messages

26. Message Options (Pin, Copy, React, Edit)

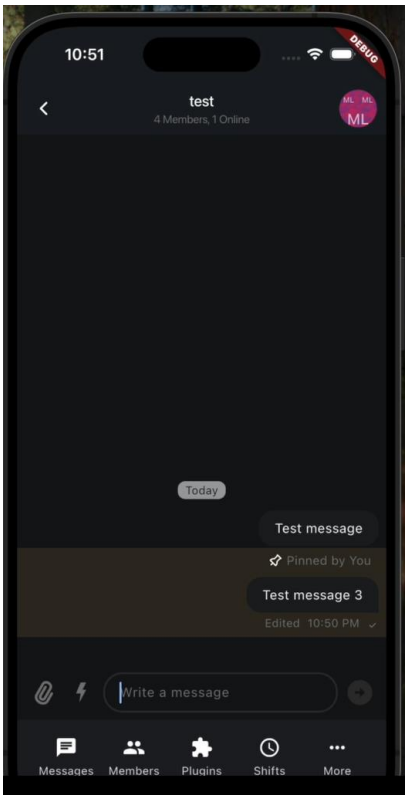


27. React to Messages

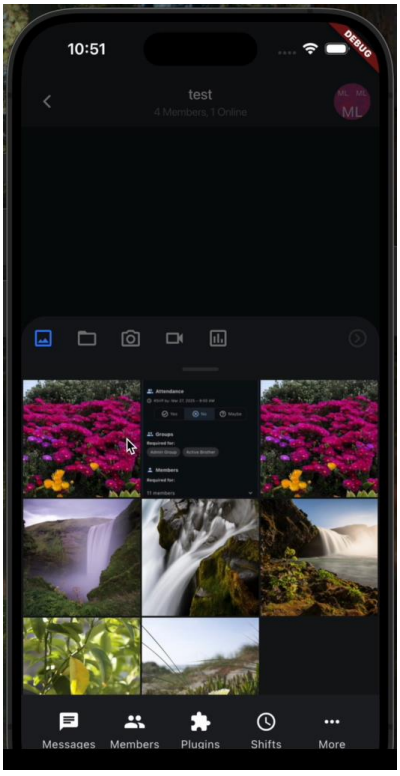


28. Edit Message

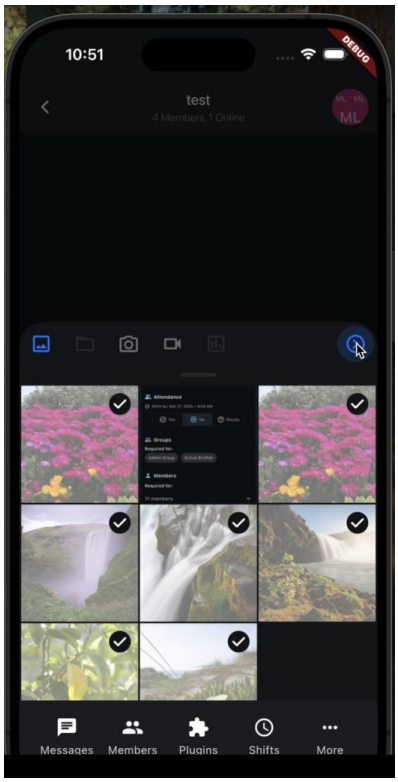
29. Pin Message



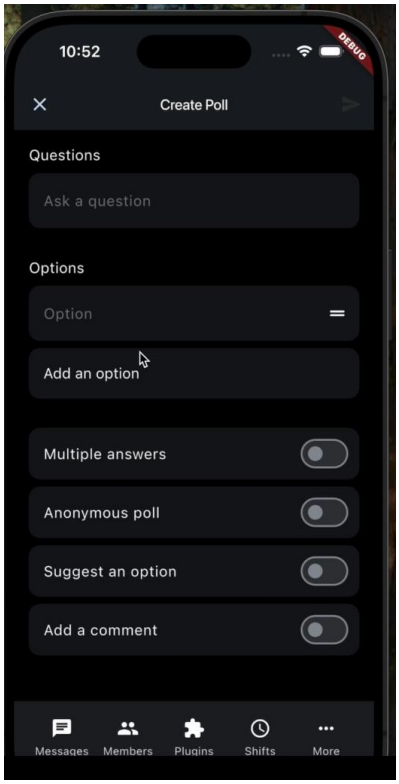
30. Add Attachment to Message



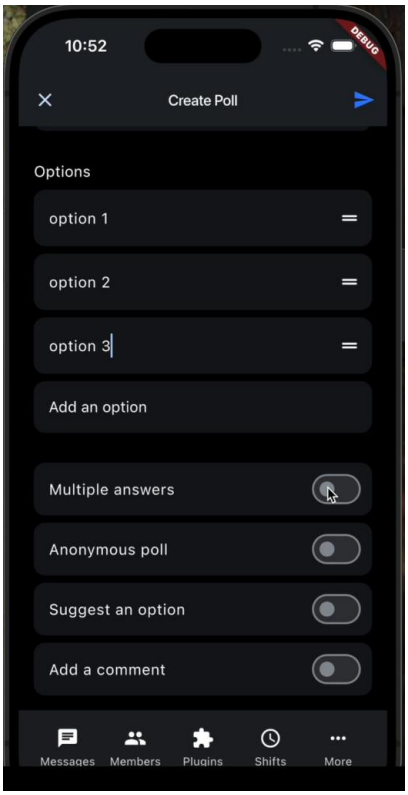
31. Select Multiple Attachments to Send



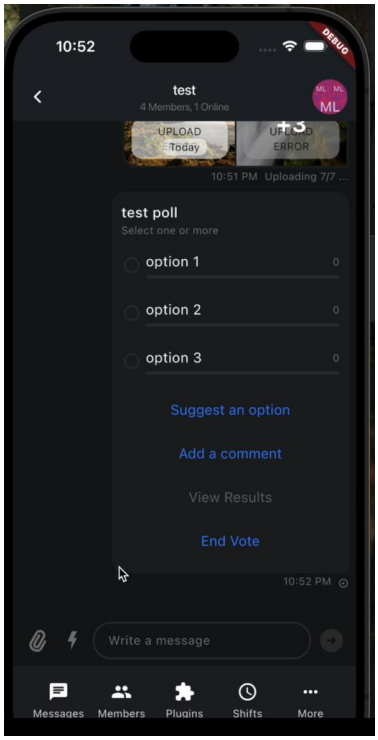
32. Create Polls



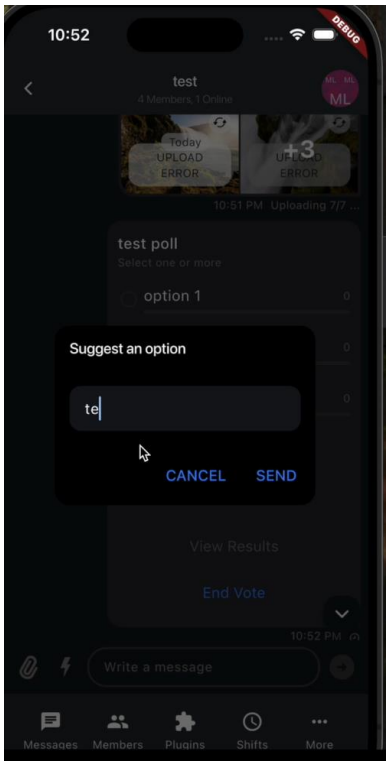
33. Customize Poll



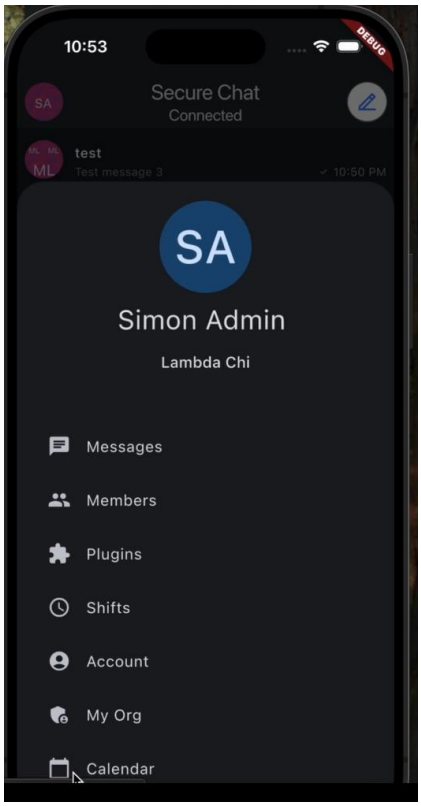
34. Send Polls



35. Suggest an option in Poll



36. My Profile Home Screen



Design

Front End Tools:

The front end of GreekRight is built using the Flutter framework, an open-source user interface framework by Google that uses Dart as its programming language. Flutter has a widget-based architecture, meaning that every element of the UI (e.g. buttons, paddings, layouts) is a widget. This allows for highly customizable, reusable, and composable components. Flutter also enables us to maintain a single codebase for mobile (Android and iOS), web, and desktop platforms which significantly reduces development time and ensures consistent user experience across devices.

Back End Tools:

The back end of GreekRight is running on Supabase, an open-source back-end platform built on PostgreSQL. Supabase provides a relational database, user authentication, file storage, and serverless functions. Each of these features is integrated with real-time capabilities and row-level security (RLS). It also auto-generates RESTful and GraphQL APIs from the database making it easy to query the data safely. Supabase automatically manages and deploys our instance to AWS for scalability and abstraction of responsibilities.

IDE:

We used the VSCode for the back and the frontend as it provided the ability to quickly switch between profiles that would enable different functionality. For example, in our front end we enabled the flutter extension that automatically handled package retrieval, IntelliSense, and debugging with minimal effort. It also allowed us to use code generators that automatically generate general functions [e.g. toJson()]. This significantly speeds up development time. For the backend, we used the Supabase CLI and extension that provided SQL IntelliSense specific to our current database state. The Supabase CLI automatically builds a completely local backend instance in docker containers which allows us to view changes with little risk, as we could revert back to the working state if necessary.

Market Competitiveness:

We believe our product is competitive in the market because it combines many of the features that our competitors offer into a single platform. Our stand-out feature is the ability to dynamically group members into specific collections. For example, many fraternities have alumni management systems that are separate from the active member attorney management system. Our system allows administrators to simply move a fraternity or sorority member into a

different group (for example, alumni), and based on that grouping, it will automatically update permissions to certain features of the app. This takes the stress out of trying to micromanage every person in an organization, and it allows the admin to just simply assign permissions to features based off their groups instead of each person. All of our features in our app use this grouping system natively. For example, in our calendar system, an admin can invite entire groups of people to an event, and as an individual's group changes, their access to that event changes. In our competitor's applications such features do not exist.

	Updated UI	Modular, Scalable Framework	Dynamic Grouping	Calendar System	Shifts System	Messaging System	Financial System
OmegaFi (1992)	✗	✗	✗	✓	✗	✓	✓
OurHouse (2014)	✓	✗	✗	✓	✗	✓	✗
Flare (2018)	✓	?	✗	✓	✗	✓	✗
GreekRight (2025)	✓	✓	✓	✓	✓	✓	Q3.25

Retrospection:

Supabase and Flutter are very powerful tools, and they both definitely have a learning curve. We solved this problem by splitting what we did for the project, mainly between the front end and the back end, allowing one of us to focus on the Flutter framework while the other one focused on the Supabase and Postgres framework. This worked out really well and allowed us to develop our app with high efficiency. Although we focused separately on different parts of the framework, we both contributed to the front end in the back end. This allowed us to understand what each of us was doing and allowed us to keep to schedule. Another problem we ran into is other schoolwork overloaded our capacity thus we switched Sprint 2 and 3, as Sprint 3 was a smaller overall feature compared to Sprint 2. This kept us on time without needing to extend our Current backlog or feel overwhelmed and behind schedule.

To improve our product, we plan to produce a financial system for money management that will allow us to monetize our user base without needing to directly charge them or show advertisements.