Updated Proposal for Orbital 25

LockBox.

Team Name	Locked In
Team ID	7100
Team Member 1	Emilia Ayu Binti Yuhasnor Affandy (Y1 Computer Science)
Team Member 2	Chow Bing Hang (Y1 Computer Science)
Advisor	Jerome Goh Zhi Sheng
Proposed Level of Achievement	Artemis

Motivation

In a world of instant gratification and fleeting moments, we strive to encourage others to pause, be present, and smell the flowers (literally) when they're travelling. Social media rewards content that is carefully curated and staged over unpolished, raw memories. However, as travel enthusiasts, we have noticed that the most meaningful memories are not those captured for public consumption, but rather the raw unfiltered experiences we laugh about years later.

Current travel and photo apps focus on storage, rather than experience. They are built for utility, and do not fully capture the essence of nostalgia. With Gen Z's desire for authenticity, we believe there is a growing need for a tool which honors the spontaneous spirit of travel.

Our app, **Lockbox**, seeks to satisfy this appetite by capturing the real, uncensored essence of group travel: the inside jokes, the awkward moments, and all those special experiences that make travelling fun. It acts as a memory capsule to let people preserve and relive the essence of travel by capturing moments as they happen, locking them away, and later unlocking them to relive them again.

Aim

We aim to create **Lockbox**, a time-capsule travel vlogging app that enables groups of friends (or solo travellers) to document their trips in a fun, interactive, and authentic way. Users can send in-app invites to friends to join a trip and contribute to a shared memory vault. Throughout the journey, instead of instantly sharing content with the whole world, users receive personalised prompts called vibe checks and respond to them by recording short clips that remain locked in a vault until a set time (such as after a trip or at the end of the day). Once the vault unlocks, everyone on the trip can access and view each other’s responses and memories - like Spotify Wrapped, but for travel memories!

User Stories

1. As a couple who wants to capture and remember special moments in our honeymoon trip, I want to be able to have a platform where we can record and store orbs that we can reminisce together on our anniversary.
2. As a solo traveller who is backpacking across the country, I want to be pushed out of my comfort zone by using prompts as a bucket-list, and finally be able to access the fun snippets of my brave adventures when I get back.
3. As a traveller, I want to record memories on my phone and store them in a locked vault for a set time, so that I can stay present in the moment and fully experience my trip without digital distractions.
4. As a group of friends on a road trip, I can react to my friends' unlocked clips, so that we can engage with each other's memories
5. As a teacher who wants my students to reflect on their learning experiences during a school trip, I want to set up educational prompts that encourage students to record their observances and compile these observances in one album.
6. As a traveller seeking a digital detox, I want to capture and store my travel memories without the constant screen time, so I can stay immersed in the experience while still preserving meaningful memories to visit later.

Features

No.	Feature (core/extension)	Description	Rough Timeline
1	User Management (core)	Lockbox currently implements a secure and user-friendly authentication system using Supabase where users can: • Sign up and login with a username and password, protected by secure encryption • Customise their profiles with profile pictures and bios In the future, users can: • Receive confirmation tokens to accept their user. • Log in with Google or Spotify accounts • Recover forgotten passwords via email • Answer an onboarding questionnaire to personalize the user experience	Liftoff
2	Add Friends (core)	Users can add friends via their unique user IDs. After becoming friends, a user who hosts can invite other friends onto trips that they organised.	Milestone 1
3	Invites (core)	Invites is a feature in which users can set up trips (feature below) and invite other friends to join a trip. <u>The host</u> The user who creates the invite and the trip. They can invite a selected list of friends to a trip. An attractive cover is chosen for the trip here, via a library of cover templates. <u>Invitees</u> Friends invited by a host. After being invited, they can decide whether to accept or decline an invitation, and accepting an invitation places the user in the list of participants for the trip.	Milestone 1
4	Trips (core)	An invite, once joined by a participant, becomes added to their dashboard as a trip. A trip is a fully interactive page which contains a capture button allowing users to capture and relive memories. Each trip page will display the trip cover, trip description, and the current list of participants who accepted the host's invite.	Milestone 1

		The features of a trip page differ depending on the host's instruction. a) <u>Before start date of the trip</u> The trip will display a countdown to the trip's commencement. An invitee viewing the invite can choose to accept or decline an invitation. b) <u>After start date of the trip, and before unlocking the Vault</u> (feature 7) The trip will display the most recent vibe check as selected by the host. A locked vault indicator will show that orbs cannot be viewed yet. In the trip menu, users can record an orb to respond to a vibe check using the capture button. Users can also use the notify button to nudge participants who haven't responded to their prompts c) <u>After unlocking the vault</u> Unlocked vault will display past orbs in channels, separated by vibe checks.	
5	Orbs (core)	Orbs are short, circular video snippets inspired by Inside Out's memory orbs and Telegram's bubble-style messages. Each orb is recorded as a response to a vibe check, and later gets stored in a shared vault until the vault unlocks after a set time. An orb is hold-to-capture, where users hold down the capture button to record the orb, and are re-recordable if users dislike their initial attempts.	Milestone 2
6	Vibe checks (core)	Vibe checks are prompts generated based on the templates of the cover and the location of the trip. The host can reshuffle from a hidden library of vibe checks, limited to three reshuffles, until they find a vibe check they wish for their trip-mates to answer. Users can also create their own vibe checks. Hosts and participants both then respond to the vibe check with an orb, which will be stored in the vibe check's respective channel in the vault.	Milestone 2
7	Vault (core)	The vault is the central memory capsule of a	Milestone 2

		trip, where orbs are stored according to vibe checks. A vault has two modes: locked and unlocked. <u>Locked mode</u> The vault will be locked for a set time initially set by the host. When locked, the trip page will indicate that the vault is inaccessible, and past orbs cannot be viewed. Users can record new orbs into the vault when the vault is locked. <u>Unlocked mode</u> When the set time is over, the vault switches to unlocked mode. The orbs will be organised and stored within specific channels corresponding to their vibe checks. Users can relive moments by scrolling down the channels.	
8	Reactions (extension)	Within the vault, participants of the same trip can react to the orbs using classic emotes such as happiness, sadness, disgust, and anger. Based on these reactions, the orbs will display corresponding auras visually, similar to the emotional effects seen in Inside Out, reflecting the mood of the orb determined by the group's collective reactions.	Milestone 3
9	Interactive map (extension)	After the trip, users can view an interactive map with all the locations marked with orbs recorded at each location. The map is stored in the unlocked vault (like a treasure map).	Milestone 3
10	AI-driven vibe checks (extension)	Hosts can input their user-generated itineraries to generate personalised vibe checks tailored to the activities of the trip.	Milestone 3
11	Gems (extension)	Gems are badges of honour given to users based on the locations they visit and the activity of the user, encouraging users to travel to less-trodden cities and attractions. These gems will be displayed on the user profile for other users to see.	Milestone 3

Timeline

Our team will adopt an AGILE workflow to break down complex tasks into manageable sprints and allow flexibility in responding to evolving user needs as we build and scale Lockbox.

The deliverables below are to be completed by their corresponding event. That said, being in the Artemis difficulty, we aim to achieve progress faster than the intended schedule.

Liftoff (Current)

During Liftoff (May 12 - May 19), our primary goal is to complete a prototype with basic user authentication and session management. During this period, we will set up our development environment, such as our Git branching strategy. We will also take the time to familiarise ourselves with our desired tech stack, namely React Native, Node.js, and Supabase.

1. User management and authentication (Feature 1)
    a. Implement sign up and login features with a username and password.
    b. Set up a user and session database for user and session management.
    c. Set up authentication mechanisms, including password hashing
    d. Feature for users to customise profile picture, name and bio.

2. Invites/Trips (Feature 3/4)
    a. Basic main page interface containing buttons for CRUD trip management
    b. Basic invite creation UI with trip details and send invite buttons
    c. Trips database
    d. Implement role-based access control (RBAC) for hosts vs participants.

Milestone 1

By Milestone 1 (June 2), we plan to have a minimal working system, with both frontend and backend integrated for the creation and management of trips and vibe checks. For media storage, we will learn how to use CloudFlare R2.

1. User management and authentication
    a. Confirmation tokens via email
    b. Sign in via Google/Spotify account
    c. Reset your password feature
    d. Onboarding questionnaire
2. Add Friends (Feature 2)
    a. Add friends feature by unique ID.
    b. Set up friends table in user database
3. Invites (Feature 3)
    a. Invite friends feature (for host)
    b. Invites table manages invitees as well as participants in the trip.
    c. Option to accept/decline an invitation.
4. Trips (Feature 4)
    a. Trip creation feature implemented by setting up an invite
    b. Minimal library of invite cover templates
    c. Trips database (frontend now interacts with backend)

Here, we will also begin setting up our testing environment, setting up our code linting tools and testing framework (eg Jest for unit tests).

1. Write unit tests for user/session logic.
2. Set up integration tests for trip creation.
3. Test full auth and invite workflows end-to-end
4. Set up issue tracking for bugs found during internal usage

Milestone 2

By Milestone 2 (June 30), we will complete a working prototype of Lockbox with all core features implemented.

1. Vibe checks (Feature 6)
    a. Vibe check generator with reshuffling feature
    b. Basic library of engaging vibe checks
    c. Creation of vibe check channels.
2. Invites (Feature 3)
    a. Template library expanded with even more templates
3. Trips (Feature 4) refinement
    a. Host option to lock/unlock vaults
4. Orbs (Feature 5)
    a. Capture button to record video when button is held down
    b. Video in circular shape
    c. Orb storage management within Supabase storage
5. Vault (Feature 7)
    a. Host ability to lock/unlock vault
    b. Vault stores orbs of different users.
    c. Vault contains channels that store all orbs within a vibe check
6. Gems (Feature 10)
    a. Collection of user activity (eg locations the user has visited, reactions user has received)

Our prototype will also be ready for external testing, and we will conduct a round of beta testing.

1. Test app across real-world devices (iOS/Android)
2. Collect UX feedback and bug reports from testers
3. Run regression tests before every sprint merge
4. Refine backend validation and error handling from test feedback
5. Validate vault/orb logic under real usage

Milestone 3

By Milestone 3 (July 28), Lockbox will evolve into a fully functional system with both core and extension features.

1. Trip/Invite (Feature 3/4)
    a. Extended library of templates.
2. Reactions (Feature 8)
    a. Ability for trip goers to react to orbs
    b. Orbs change colour depending on reaction
3. Interactive Map (Feature 9)
    a. Location tracking of user and orb when the orb is recorded.
    b. Integration of map API to visualise locations
    c. Pins and markers to display where each user has been
    d. Secure location storage for tracking where user has visited
    e. Map is displayed in an unlocked vault.
4. AI-driven vibe checks (Feature 10)

    a. Generate vibe checks based on prompts by user, which include user itineraries and key words.
    b. Vibe checks can be stored in vibe check library
5. Gems (Feature 11)
    a. Design of gems
    b. Users can earn gems after completing certain criteria.
    c. Display gems on user profile once acquired.

We will conduct an extensive round of beta testing here.
1. Extensive UI/UX testing across edge cases (vault unlocking, vibe check flow, reactions)
2. Stress test media upload/download
3. Final regression + acceptance testing
4. Collect final bug reports and feature feedback for post-launch iteration

Tech Stack

1. React Native (frontend)

We chose React Native as it supports rapid development of a cross-platform mobile app from a single codebase. Its strong ecosystem and built-in support for animations, gestures, and community libraries (like Expo and NativeWind) make it ideal for building Lockbox, as it will incorporate a lot of highly interactive and visually engaging elements like Orbs.

2. Node.js, Express.js (backend)

Node.js with Express.js gives us fine-grained control over backend logic and allows us to build custom APIs tailored to our app's evolving needs. This setup gives us flexibility beyond what Supabase's client SDK offers, especially for things like access control, media uploads, session handling, and advanced business logic. We will use tools like Multer for image uploading.

3. Supabase (database)

We chose to use Supabase as a scalable, open-source PostgreSQL backend with built-in authentication, real-time updates, and row-level security (RLS). Its tight integration with our frontend and ease of setup allows us to build fast while maintaining control over our data and permissions.

4. Cloudflare R2 (video and photo storage and CDN)

Cloudflare R2 is a cost-effective, S3-compatible object storage with no egress fees, which is ideal for storing and delivering our Orbs and trip thumbnail templates. Its integration with Cloudflare's global CDN ensures our content loads quickly and reliably, regardless of where users are.

5. Mapbox (Map API)

Mapbox provides a highly customizable and visually appealing map experience, essential for location-based features like setting trip destinations and visualizing memories. Its offline support and flexible SDK make it more adaptable than alternatives like Google Maps for our travel-focused use case.

6. OpenAI API (AI-driven prompt generator)

We will use the OpenAI API to power our vibe checks. By inputting the user's generated itineraries, the OpenAI API will analyse the overall tone and mood of the trip and return a short vibe check that captures the trip's "vibe".

Qualifications

Technical Skills

- Foundation in programming and software development from relevant CS courses (CS1101S, CS2030S, CS2040S).
- Proficient in Python, Java, and JavaScript; familiar with TypeScript; highly proficient in SQL
- Mobile app development using Swift.
- Web development experience with Flask and Django.
- Basic understanding of React.js, Ruby on Rails, Go and AWS services.

Project Experience

- Developed a mobile app, Bubbles, using Swift during Hack & Roll 2025.
- Self-taught React.js and Ruby on Rails under the CVWO Winter Assignment.
- Built web applications using Flask and Django via CS50 courses, with projects such as a mailing system, an auctioning web app, and a note-taking app.
- Experience with software development in extracurricular activities (Computing and Robotics Club).
- Applied object-oriented programming and data structures in various coding projects.

Software Engineering

Software Engineering Practices

We will implement certain industry-standard software engineering practices throughout development to ensure that Lockbox is robust, scalable, and maintainable.

Version Control
We will use Git and Github for version control. Each feature and enhancement will be developed in its own branch before being merged into the main branch via pull requests. This ensures better collaboration, code review, and maintainability.

Quality Checks
We will implement rigorous quality checks by writing comprehensive unit tests and integration tests to validate the correctness and functionality of individual components as well as their interaction within the system.

1. We will use Jest for unit testing and the React Native Testing Library for component-level testing on the frontend. On the backend, Supertest is used to test our Express.js APIs.
2. Additionally, we will conduct thorough type checks through TypeScript to ensure type safety and prevent potential runtime errors. These tests will be executed and peer reviewed by each other before merging any changes into the main branch, ensuring that the code remains reliable.
3. We will use ESLint and Prettier to enforce code style consistency and prevent common bugs before runtime.

AGILE sprints
We will adopt the industry standard two-week sprints for iterative development to ensure steady development pace and ensure organised workflows. Attaining regular feedback allows us to be adaptable to changing requirements.

CI/CD

We will implement CI/CD pipelines using GitHub Actions to streamline the software development lifecycle and automate our development and deployment workflows.

1. Continuous Integration (CI) refers to the practice of automatically and frequently integrating code changes into a shared source code repository. On every push or pull request, GitHub Actions will automatically run our test suite and check for type errors through the quality checks listed above. This will allow early detection of issues.
2. Continuous Development (CD) after passing tests in the CI pipeline. Our frontend build is built through Expo, and will be deployed to Expo Go using GitHub Actions and EAS (Expo Application Services). Our backend will be deployed to a cloud provider (such as Vercel) for staging and production. This setup ensures that software can be pushed out frequently and reliably, with minimal manual intervention

Software Engineering Principles

We will apply software engineering principles to guide how we structure our code logic, APIs, and system architecture.

SOLID principles

By leveraging OOP concepts like inheritance and polymorphism, we will reinforce key SOLID principles to enhance code readability, reusability and maintainability. Some examples are below:

1. S - Single Responsibility Principle
    a. Each class should only have one specific responsibility.
        i. Invite Manager - Manages invite creation
        ii. Trip Manager - Manages trip creation and updates
        iii. Vibe Check Generator - generates vibe checks
2. O - Open-Closed Principle
    a. Implement separate feature modules for features in a trip such as participant management, vault unlocking, vibe checks that can be attached dynamically to a trip.
3. I - Interface Segregation Principle
    a. The frontend should only call APIs that return the data that it needs.
    b. We will split the APIs into smaller, purpose-driven APIs:
        i. Instead of retrieving all user details, only retrieve Profile API to retrieve user profile data (eg past trips, gems earned)
        ii. Instead of retrieving all trips, only return trips the user is involved in.
4. D - Dependency Inversion Principle
    a. Abstractions should not depend on details, but details should depend on abstractions.
    b. Lockbox will be dependent on Supabase for the database. In the event we run into trouble with Supabase, abstractions prevent dependencies on Supabase for Lockbox's core logic.
        i. Create an abstract interface that works with any database
        ii. Implement a Supabase repository.
        iii. Use dependency injection to switch between repositories in the event we want to use other tech stacks over Supabase.

Principle of Least Privilege (PoLP)

We will adhere to the PoLP principle by using Role-Based Access Control (RBAC) to ensure certain trip management features are restricted to hosts, such as unlocking a vault or selecting a vibe check.

DRY principle

We will adhere to the DRY principle (Don't Repeat Yourself) prudently, ensuring that any modifications or fixes to a specific feature can be seamlessly integrated without necessitating extensive refactoring across the codebase.