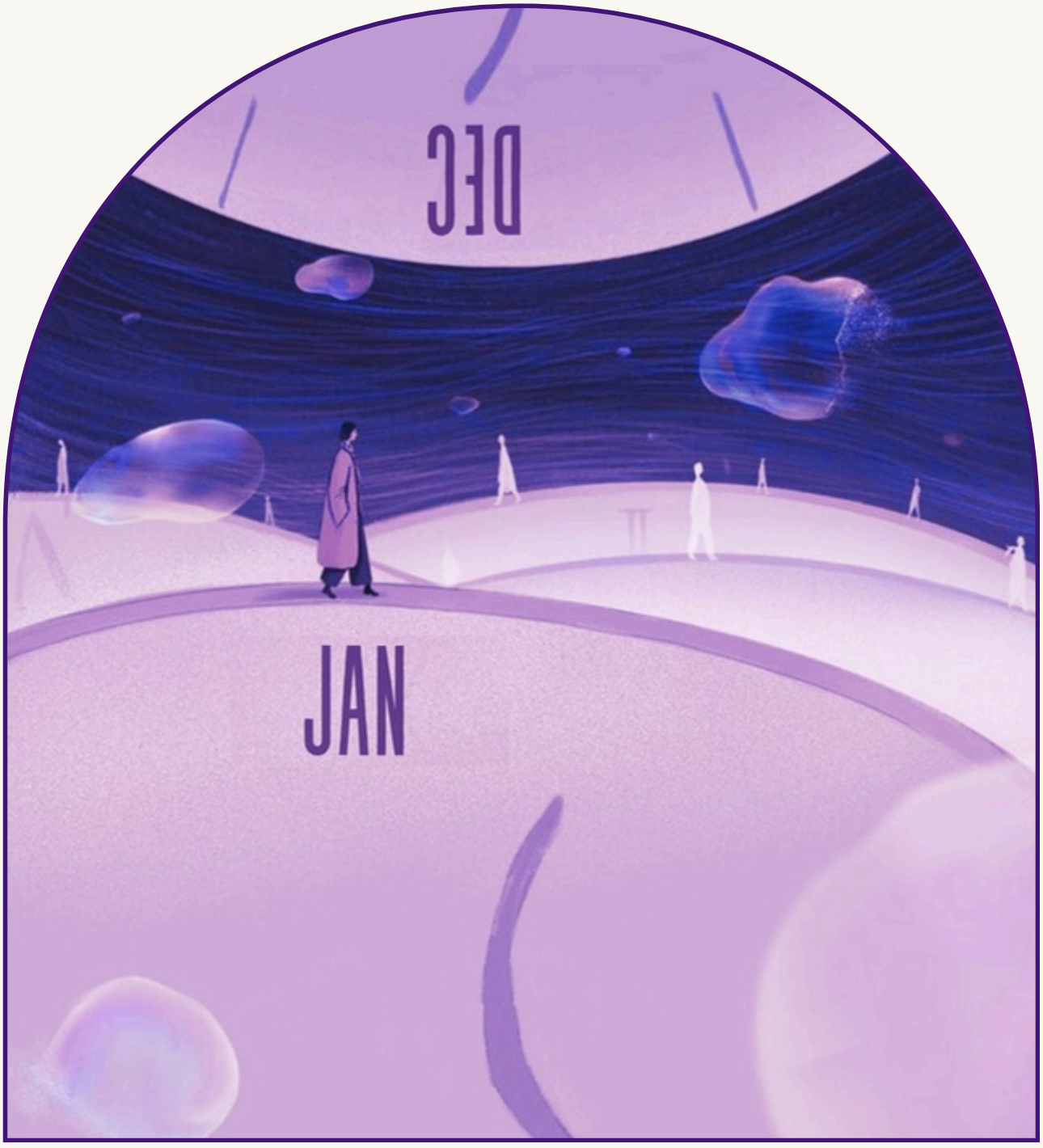
SPHERIS 24/7 FINAL



OWN YOUR TIME, SHAPE YOUR FUTURE.

TABLE OF CONTENTS

SPHERIS 24/7

1.1 SHORT DESCRIPTION

In today's busy world, managing multiple tasks and staying focused on long-term goals can be challenging. People often find themselves juggling between work, study, personal commitments, and various deadlines, which can lead to feelings of stress and disorganization. Spheris 24/7 is designed to address these challenges by offering a simple, all-in-one platform that helps users plan, organize, and track their goals and tasks effectively.

With a range of features like personalized reminders, progress tracking, and a customizable interface, Spheris 24/7 aims to provide users with the tools they need to stay organized, prioritize tasks, and maintain focus. Whether managing daily responsibilities, long-term goals, or personal growth, the application empowers users to be more productive and achieve their objectives with ease. Through its intuitive design and user-centered approach, Spheris 24/7 makes it easier to stay on top of tasks, reduce stress, and make the most of every day.

SPHERIS 24/7

1.2 CUSTOMER STATEMENT OF REQUIREMENTS

SPHERIS 24/7

PROBLEM OR STATMENT

While many try to develop time management skills, a plethora of barriers makes it an impossible task. Some have the requisite skills to manage their activities but lack the discipline to follow through because of their busy schedules, while others have problems with goal setting or progress tracking. It is not unusual for many to feel stressed, unproductive, and unaccomplished.

A major challenge is task overload and poor prioritization. With numerous deadlines competing for attention, it's easy to focus on urgent tasks while neglecting long-term goals. Without a clear system for ranking priorities, productivity suffers, and key objectives remain unfulfilled.

Constant distractions also make it difficult to maintain focus. Digital notifications, social media, and multitasking pull attention in multiple directions. Many users rely on various productivity apps, but switching between multiple tools often leads to scattered information and inefficiency rather than clarity.

Additionally, lack of structure and accountability prevents individuals from breaking down goals into actionable steps and tracking progress. Without a way to build consistency, motivation fades, and ambitions remain unrealized. Finally, maintaining a healthy work-life balance has become increasingly difficult. Many people fall into unsustainable routines, struggling to make time for personal growth and well-being. Poor time management leads to burnout, reduced productivity, and an overall decline in quality of life.

To address these challenges, Spheres 24/7 is designed to simplify this process by providing a comprehensive, platform that streamlines planning, organization, and goal tracking. By integrating smart reminders, progress monitoring, and a fully customizable interface, empowering users to take control of their time, focus on what matters, and achieve their goals with greater clarity and ease.

ABOUT SPHERIS

SCOPE

24/7

THE APP IS DESIGNED FOR:

- STUDENTS WHO NEED TO MANAGE ASSIGNMENTS, DEADLINES, AND STUDY SCHEDULES EFFECTIVELY.
- WORKING PROFESSIONALS LOOKING TO OPTIMIZE THEIR PRODUCTIVITY, MANAGE PROJECTS, AND STAY ON TOP OF DEADLINES.
- SELF-IMPROVEMENT ENTHUSIASTS SETTING PERSONAL GOALS FOR FITNESS, READING, OR SKILL DEVELOPMENT.
- PROJECT MANAGERS & TEAMS WHO REQUIRE STRUCTURED PLANNING, TASK DELEGATION, AND PROGRESS TRACKING.
- NEURODIVERGENT INDIVIDUALS (E.G., ADHD) WHO BENEFIT FROM STRUCTURED ORGANIZATION, REMINDERS, AND FOCUS TOOLS.
- CREATIVES & CONTENT CREATORS MANAGING PROJECTS, DEADLINES, AND IDEA TRACKING EFFICIENTLY.

OBJECTIVES

Our application aims to enhance productivity and organization by providing users with a structured approach to goal-setting and daily planning. The app utilizes AI-powered task management, smart reminders, and a customizable dashboard to help users efficiently manage their time and responsibilities. Our Services:

1. Yearly & Monthly Goal Structuring: Users can define long-term goals and break them down into manageable monthly and daily tasks, ensuring consistent progress toward their objectives.

2. Daily Planner Sections:
The app includes a to-do list, journaling space, a daily timeline, and a progress tracker, allowing users to plan their days effectively.

3. Task Prioritization:
With AI-powered analysis, the app ranks tasks based on urgency, importance, and user behavior, helping users focus on what matters most.

4. Smart Reminders & Notifications:
The app sends personalized reminders based on deadlines, missed tasks, or habits that require attention, keeping users on track.

5. Study & Work Timer (Pomodoro Mode):
An integrated timer enables users to work in focused sessions with scheduled breaks, improving productivity and time management

6. Customizable Dashboard:
Users can personalize their planner layout by selecting which tools (to-do list, calendar, progress tracker, etc.) appear on their home screen, making the experience tailored to their needs.

7. Task Categorization Tool:
This feature allows users to organize their tasks into customizable categories based on different aspects of their life, ensuring a more structured and efficient planning experience.

8. Rewards System:
Users can earn points, badges, or other incentives for completing tasks, maintaining streaks, and achieving goals, making productivity more engaging and motivating.

9. Complete Tasks with Others:
The app allows users to collaborate on tasks with friends, colleagues, or study partners. Users can assign shared tasks, track progress together, and support each other in completing goals, fostering teamwork and accountability.

RELATED WORK

TickTick :
offers a comprehensive solution for task management, providing goal structuring, AI-powered task prioritization, habit tracking, and a Pomodoro timer. It allows users to break down long-term goals into smaller, manageable tasks, improving cognitive efficiency. Additionally, TickTick's habit-tracking and reward system encourages consistency and motivation. However, it faces criticism for its overwhelming interface, especially for new users, and its lack of advanced collaboration tools, which limits its use for team-based projects.[1]

Notion :
offers the highest level of customization among these apps. Users can create personalized workflows that combine goal-setting, journaling, and task categorization in a flexible layout. This flexibility supports cognitive ergonomics, as customizable tools can enhance user satisfaction and efficiency. However, Notion's steep learning curve and reliance on manual organization may deter some users, especially those who prefer a more automated and ready-to-use solution. It also lacks native AI-powered task prioritization, which could limit its ability to streamline task management for users seeking efficiency.[3]

Todoist :
is another widely-used app, renowned for its AI-driven task prioritization and seamless collaboration features. It ranks tasks based on urgency and importance, which helps users focus on what matters most. Todoist also integrates with third-party tools, making it a strong option for both personal and team productivity. However, it does not provide detailed insights into long-term goal tracking, and its reward system lacks depth, which can reduce user engagement over time.[2]

02. GLOSSARY OF TERMS

TERM	DESCRIPTION
USER	An individual who seeks to organize and manage their tasks, schedules, and responsibilities efficiently, value structure, productivity, and streamlined workflows, often looking for tools that help them track progress, set priorities, and stay on top of deadlines. These users may range from professionals managing work projects to students keeping up with assignments or individuals planning personal tasks and goals.
STAKEHOLDERS	Stakeholders are individuals, groups, or organizations that have an interest in a particular project, system, or business and are affected by or can affect the outcomes of that project. They can be involved in various ways, such as decision-making, funding, or being impacted by the results.
FUNCTIONAL REQUIREMENTS	Functional requirements specify what a system must do, outlining its behaviors and features. They describe the interactions between the system and users, or other systems. These requirements focus on the core functionality needed to meet user needs and business objectives.
NON-FUNCTIONAL REQUIREMENTS	Non-functional requirements define how a system performs its functions, focusing on qualities like performance, security, and usability. They outline constraints or criteria that the system must meet, such as response time, reliability, or scalability. These requirements ensure the system's overall quality and user satisfaction
USE CASE	A use case defines how a user interacts with a system to achieve a specific goal. It outlines the steps taken by both the user and the system, including possible alternative flows.
SPHERIS	The tasks view in Spheris 24/7, where users can organize, track, and manage their tasks efficiently. It serves as a central hub for planning and prioritizing daily responsibilities, deadlines, and progress.
ORBIT GARDEN	A central feature in the application that represents the user's progress and activities in a cosmic-themed environment. It consists of three interconnected pages: Planetarium, Galaxy Squad, and Stellar Progress, each providing a unique perspective on the user's journey and engagement

02. GLOSSARY OF TERMS

TERM	DESCRIPTION
PLANETARIUM	A visual representation of the digital galaxy, displaying planets, and their names. It serves as an interactive or view-only feature where users can explore their personal universe and track their evolving journey
GALAXY SQUAD	A social hub within the Orbit Garden where users can view their group, interact with members, and monitor shared goals or deadlines. It fosters collaboration and community-building by visually representing each member's contributions and progress
STELLAR PROGRESS	A visual tracker within the Orbit Garden that represents a user's journey through different planets based on their progress, habits, and achievements. It displays a stellar map with planets the user has visited, each representing milestones or accomplishments. The Journey Log provides details such as the current planet, time logged, and the next destination. This feature helps users monitor their growth, stay motivated, and engage with their personal development in a gamified space-themed environment
SEQUENCE DIAGRAM	Shows object interactions arranged in time sequence
STRUCTURAL DIAGRAM	A conceptual modeling tool that make up a system such as a database or an application
UNIT TESTING	A level of software testing where individual units/components of a software are tested
INTEGRATION TESTING	A level of software testing where individual units are combined and tested as a group
ACCEPTANCE TESTING	A level of software testing where a system is tested for acceptability

3.1 FUNCTIONAL REQUIREMENTS

REQ1	THE USER SHALL BE ABLE TO CREATE ACCOUNT BY PROVIDING (USERNAME, EMAIL, PASSWORD)
REQ2	THE USER SHALL BE ABLE TO LOG-IN BY PROVIDING (USERNAME, PASSWORD)
REQ3	THE USER SHALLBE ABLE TO LOG-OUT.
REQ4	THE USE SHALL BE ABLE TO CUSTOMIZE THEIR VIEW OF (SPHERE, ORBIT GARDEN, CALENDAR, JOURNAL, THEME (COLOR, FONT)).
REQ5	THE USER SHALL BE ABLE TO INITIATE JOURNAL BY PROVIDING (TITLE)
REQ6	THE USER SHALL BE ABLE TO ADD CONTENT TO THE JOURNAL BY THE TOOLS(TEXT, PENCIL, HIGHLIGHTER, IMAGE , STICKY NOTE)
REQ7	THE USER SHALL BE ABLE TO EDIT JOURNAL.
REQ8	THE USER SHALL BE ABLE TO CREATE NEW SPHERIS BY ADDING (TITLE)
REQ9	THE USER SHALL BE ABLE TO EDIT SPHERIS (TITLE).
REQ10	THE USER SHALL BE ABLE TO DELETE SPHERIS.
REQ11	THE USER SHALL BE ABLE TO ADD A GOAL BY ITS NAME.

3.1 FUNCTIONAL REQUIREMENTS

REQ12	THE USER SHALL BE ABLE TO EDIT THEIR GOAL.
REQ13	THE USER SHALL BE ABLE TO DELETE THEIR GOAL.
REQ14	THE USER SHALL BE ABLE TO ADD A TASK WITH ADDING (SPHERIS NAME, TASK TITLE, NOTES, REMINDER, DUE DATE, FILE ATTACHMENT, AND CUSTOM REPEAT OPTIONS)
REC15	THE USER SHALL BE ABLE TO EDIT ANY COMPONENT OF TASKS INCLUDING(TASK TITLE, NOTES AREA, REMINDER, DUE DATE, FILE ATTACHMENT, AND CUSTOM REPEAT OPTIONS)
REQ16	THE SYSTEM SHOULD REMIND THE USER OF TASK WITH SPECIAL RESTRICTIONS (REMIND ME, DEADLINE, URGENT) BY EMAIL.
REQ17	THE SYSTEM SHOULD CALCULATE THE PROGRESS OF COMPLETED TASKS FROM THE USER'S TASKS.
REQ18	THE USER SHALL BE ABLE TO DELETE TASK.
REQ19	THE USER SHALL BE ABLE TO DELETE TASK.
REQ20	THE SYSTEM SHOULD DISPLAY TIMELINE OF THE TASK.
REQ21	THE SYSTEM SHALL ALLOW USERS TO TOGGLE YEARLY, MONTHLY AND DAILY CALENDAR VIEW

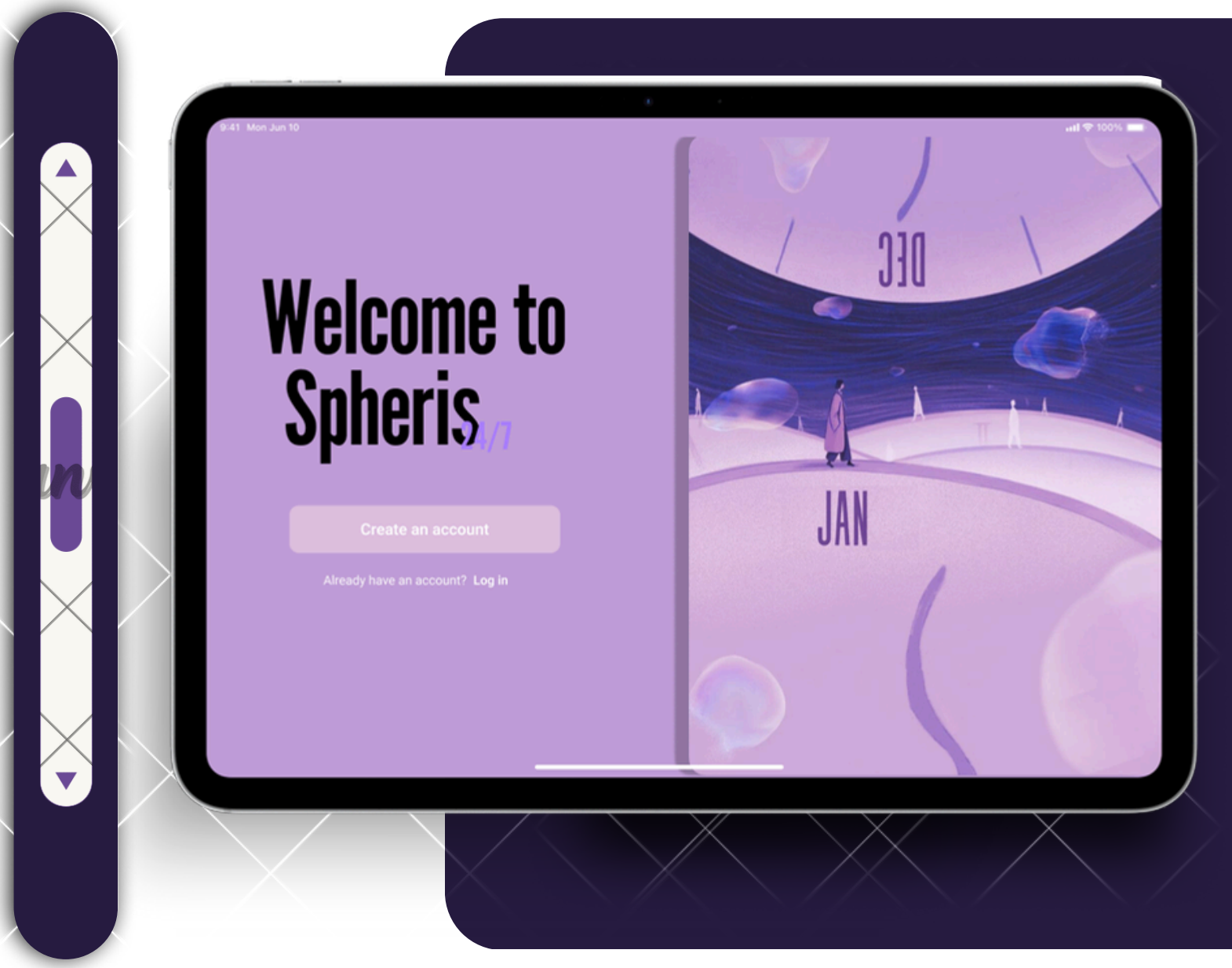
3.1 FUNCTIONAL REQUIREMENTS

REQ22	THE SYSTEM SHALL ALLOW USERS A CALENDAR VIEW OF ALL TASKS, AND SPHERIS AS A CATEGORIES
REQ23	THE USER SHALL BE ABLE TO TRAVEL THROUGH PLANETARIUM BY INCREASING THE NUMBER OF WORKING HOURS.
REQ24	THE USER SHALL BE ABLE TO VIEW THE PLANETARIUM.
REQ25	THE USER SHALL BE ABLE TO VIEW THE GALAXY SQUAD GROUP AND ALL ASSOCIATED DEADLINES.
REQ26	THE USER SHALL BE ABLE TO SEARCH FOR ANY USERNAME WITHIN THE GROUP.
REQ27	THE SYSTEM SHOULD DISPLAY THE PROGRESS STELLAR MAP WITH THE JOURNAL DETAILS
REQ28	THE USER SHALL BE ABLE TO VIEW THEIR CURRENT PLANET
REQ29	THE USER SHALL BE ABLE TO VIEW THEIR (TRAVELER ID (USERNAME), DESTINATION, TIME LOGGED, PLANT DESCRIPTION, NEXT STOP)
REQ30	THE USER SHALL BE ABLE TO UPDATE THEIR ACCOUNT DETAILS.
REQ31	THE USER SHALL BE ABLE TO DELETE THEIR ACCOUNT.
REQ32	THE SYSTEM SHOULD PROVIDE HELP THAT OFFER GUIDANCE ON USING FEATURES AND ANSWERING FREQUENTLY ASKED QUESTIONS.

3.2 NON FUNCTIONAL REQUIREMENTS

REQ33	SPEED: THE SYSTEM SHOULD ENSURE A RESPONSE TIME OF LESS THAN 2 SECONDS.
REQ34	SECURITY: THE SYSTEM SHOULD BE ROBUSTLY SECURE TO PREVENT ANY EXTERNAL ATTACKS.
REQ35	AVAILABILITY: THE SYSTEM SHALL BE AVAILABLE 98% OF THE TIME
REQ36	SIZE: SYSTEM SHOULD NOT EXCEED 100MB ON A DEVICE WHILE EFFICIENTLY MANAGING DATA CACHING AND SYNCHRONIZATION
REQ37	ESE OF USE: THE SYSTEM SHOULD HAVE AN INTUITIVE AND USER-FRIENDLY INTERFACE, ENSURING THAT USERS CAN NAVIGATE AND MANAGE TASKS WITH MINIMAL LEARNING EFFORT AND COMPLETE CORE ACTIONS IN THREE STEPS OR FEWER.
REQ38	COMPATIBILITY: THE SYSTEM SHOULD BE COMPATIBLE WITH ANDROID, IOS, AND WEB BROWSERS, ENSURING SEAMLESS FUNCTIONALITY ACROSS DIFFERENT SCREEN SIZES AND OPERATING SYSTEM VERSIONS (IOS , ANDROID)
REQ39	BACKUP: SPHERIS SHOULD AUTOMATICALLY SYNC AND BACK UP DATA AUTOMATICALLY FOR SEAMLESS RECOVERY.
REQ40	MEAN TIME BETWEEN FAILURES (MTBF) OF THE SYSTEM SHALL BE AT LEAST 30 DAYS.
REQ41	THE TIME REQUIRED TO DETECT, REPAIR AND RECOVER ALL DATA SHALL NOT EXCEED 20 MINUTES

3.3 ON-SCREEN APPEARANCE REQUIREMENTS

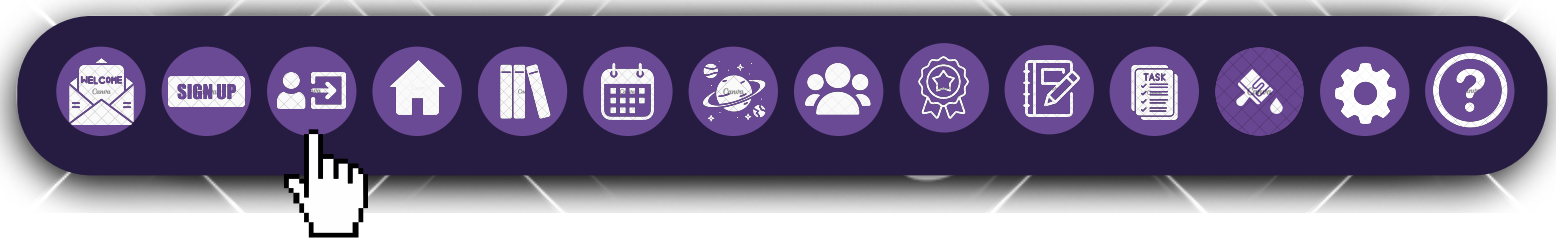


SPHERIS 24/7

3.3 ON-SCREEN APPEARANCE REQUIREMENTS

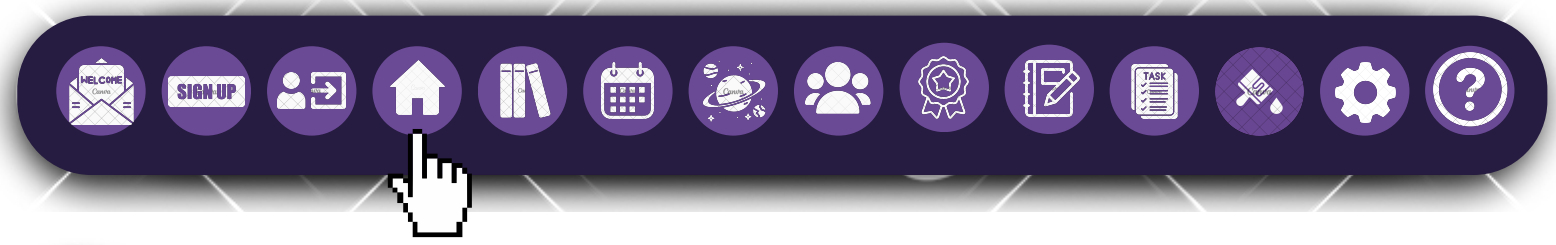
SPHERIS 24/7

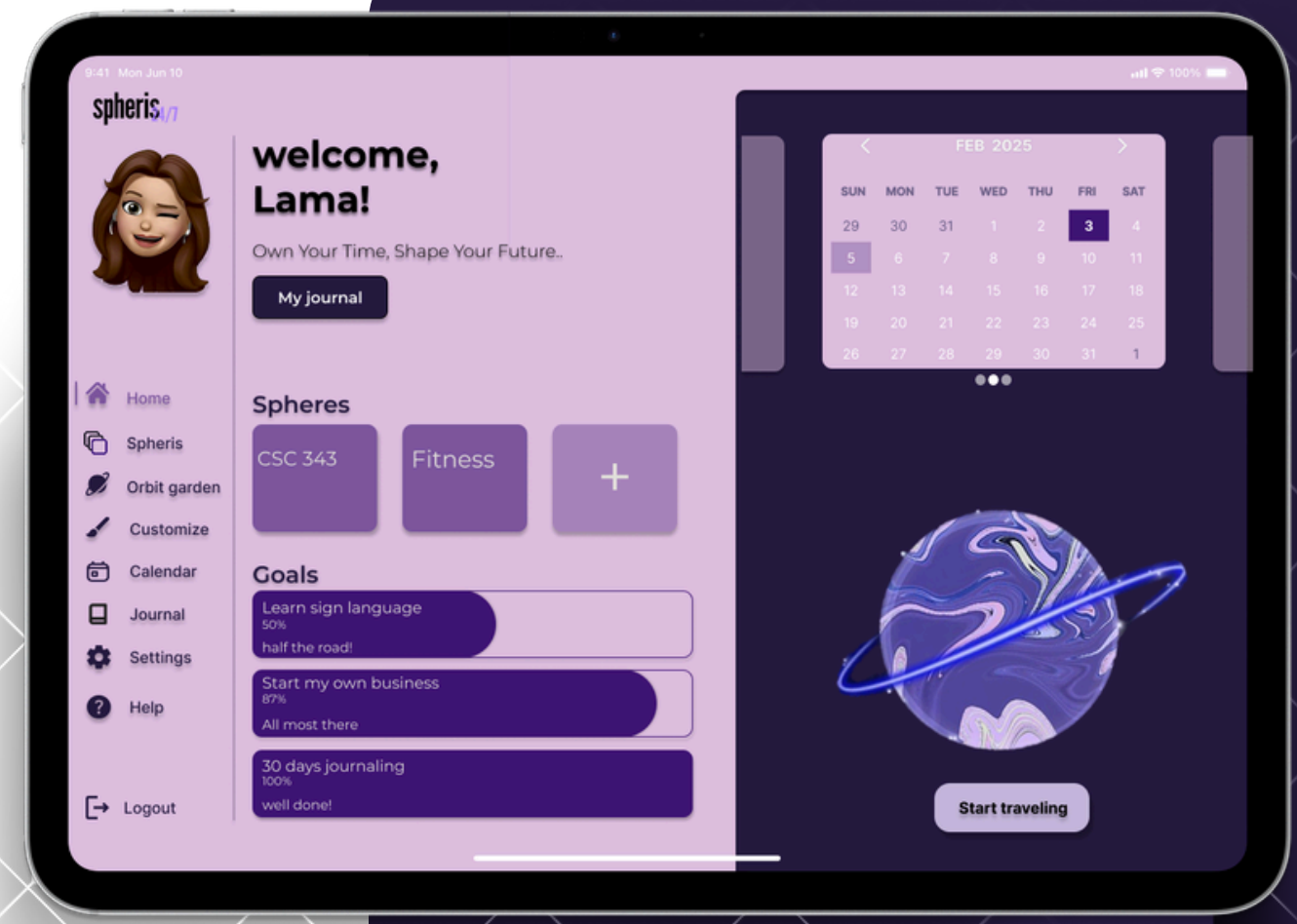
3.3 ON-SCREEN APPEARANCE REQUIREMENTS



SPHERIS 24/7

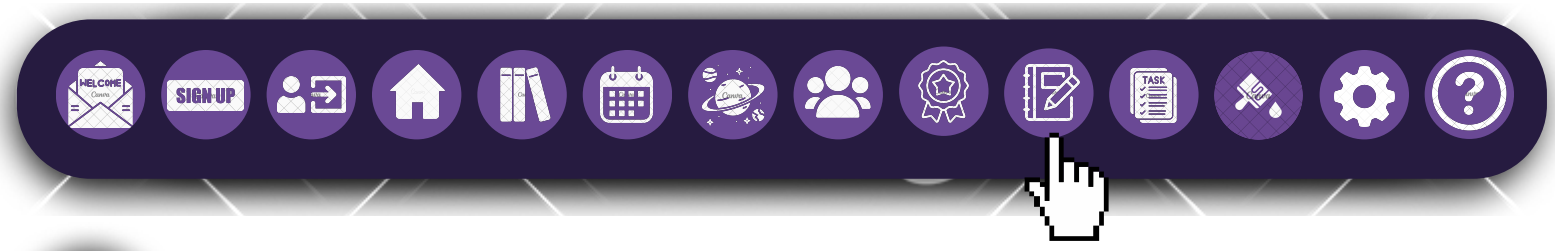
3.3 ON-SCREEN APPEARANCE REQUIREMENTS

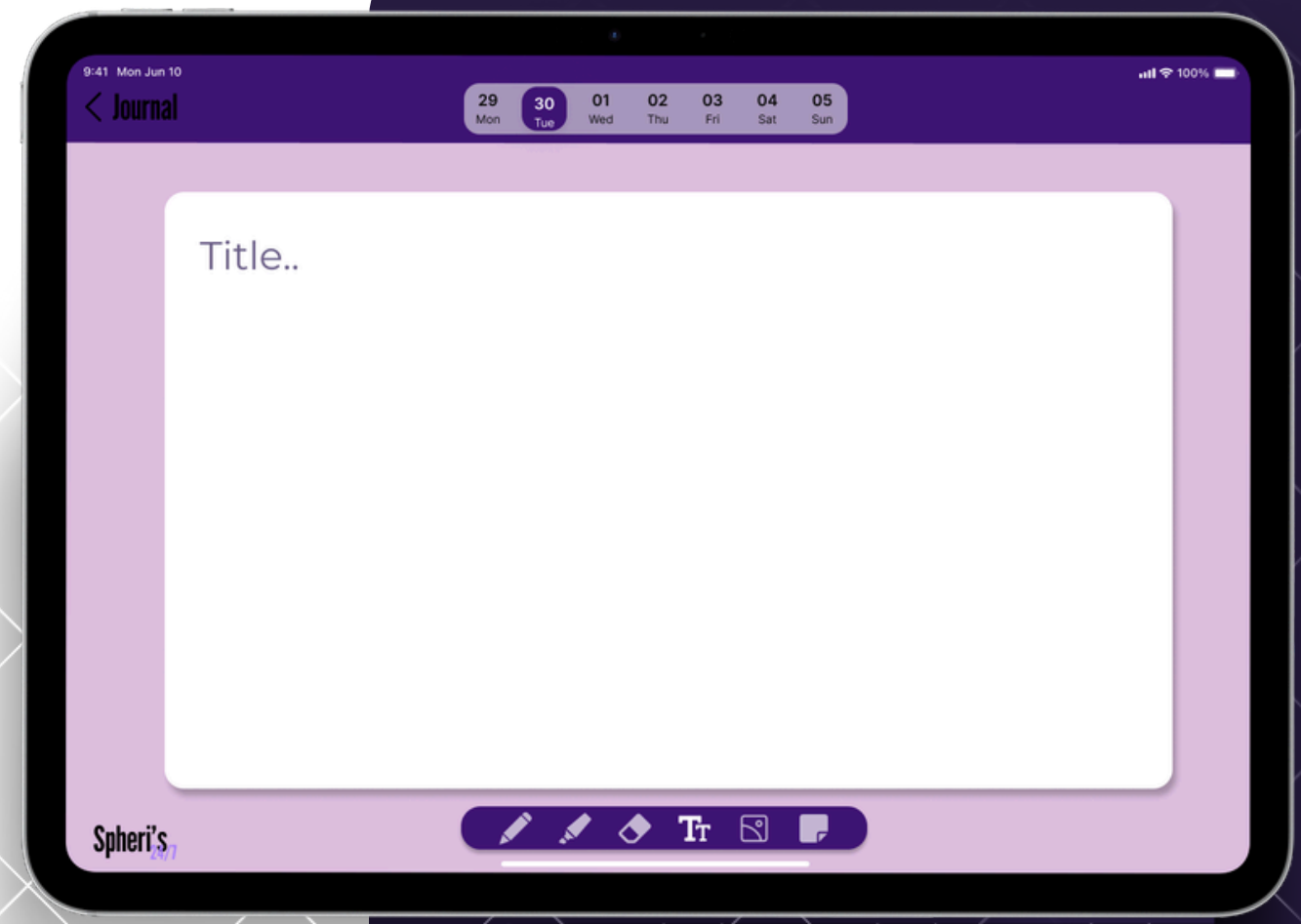




SPHERIS 24/7

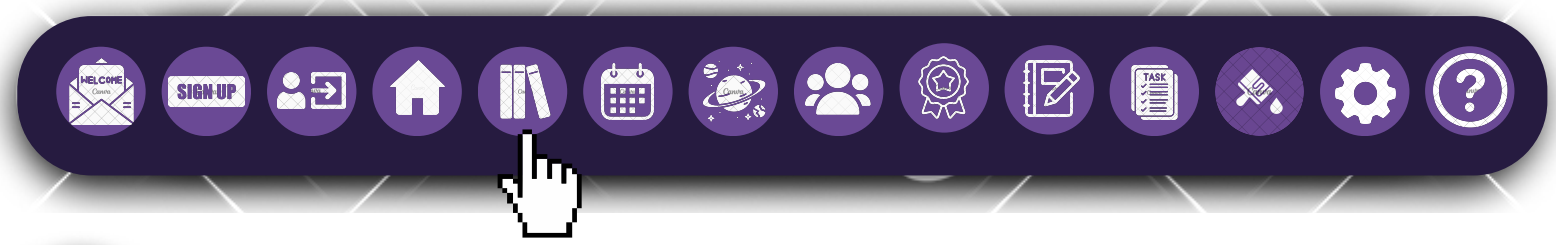
3.3 ON-SCREEN APPEARANCE REQUIREMENTS





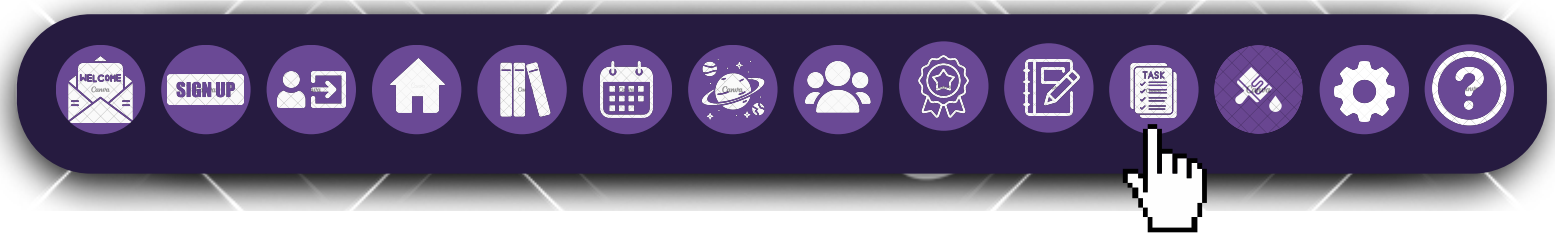
SPHERIS 24/7

3.3 ON-SCREEN APPEARANCE REQUIREMENTS



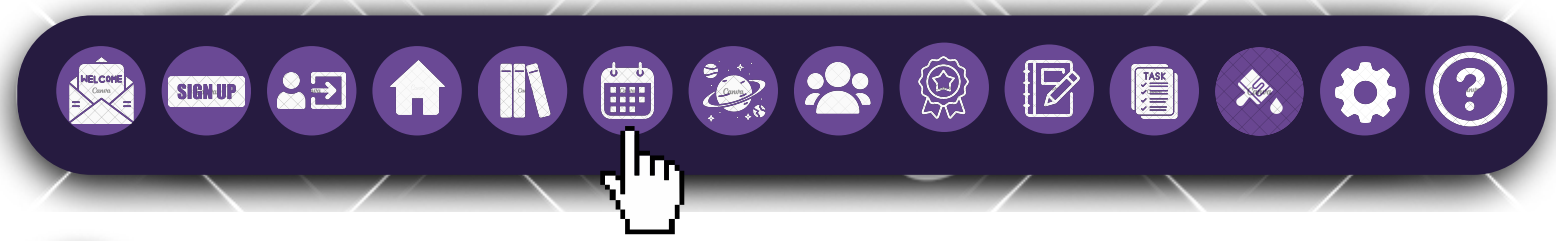
SPHERIS 24/7

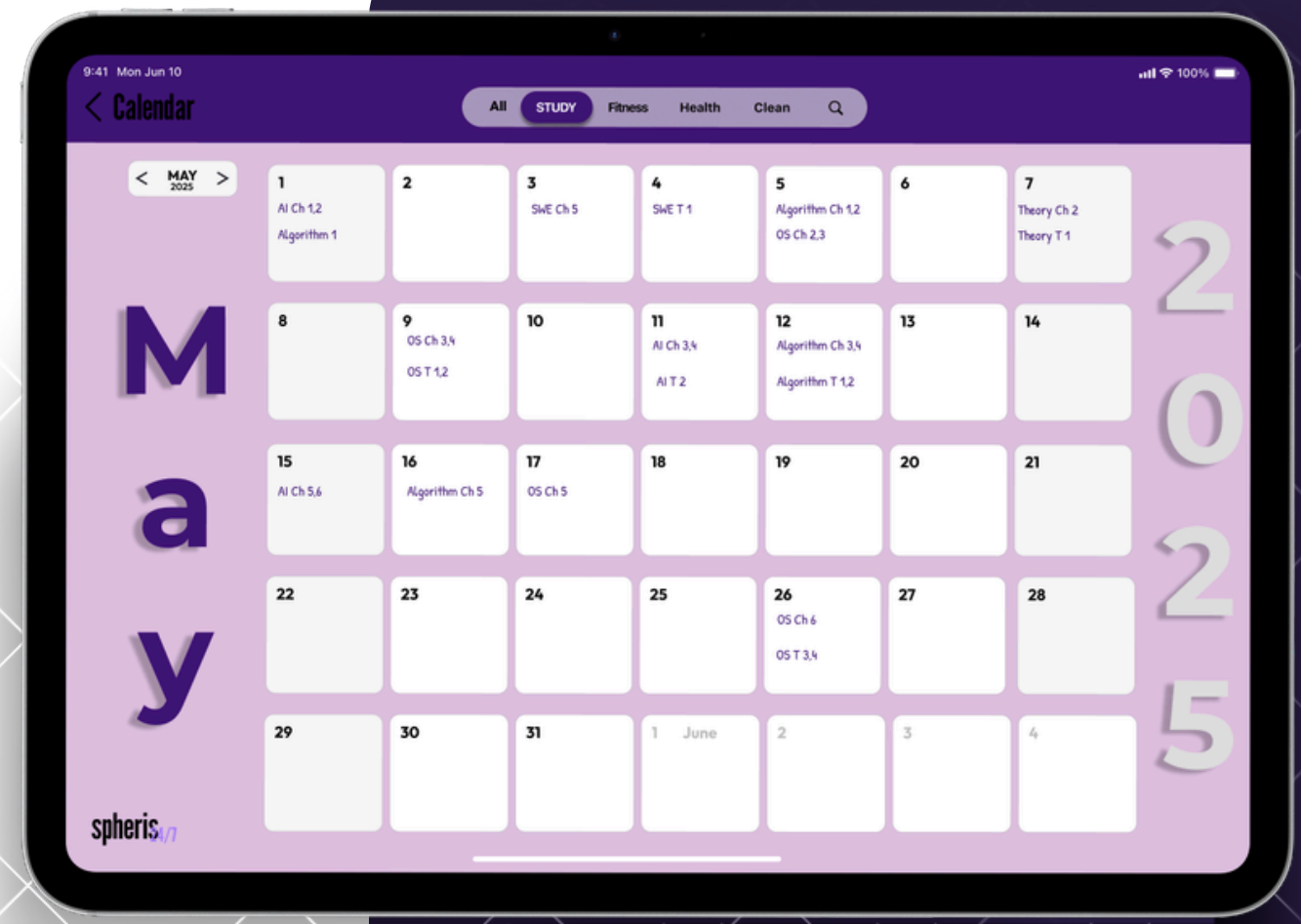
3.3 ON-SCREEN APPEARANCE REQUIREMENTS



SPHERIS 24/7

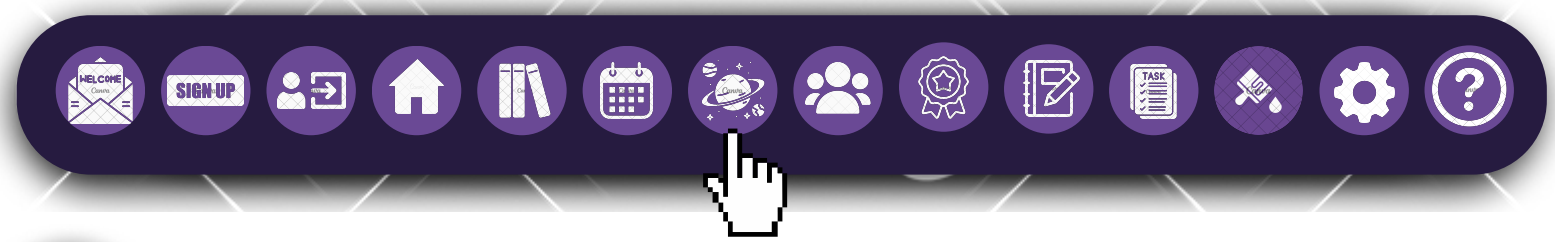
3.3 ON-SCREEN APPEARANCE REQUIREMENTS

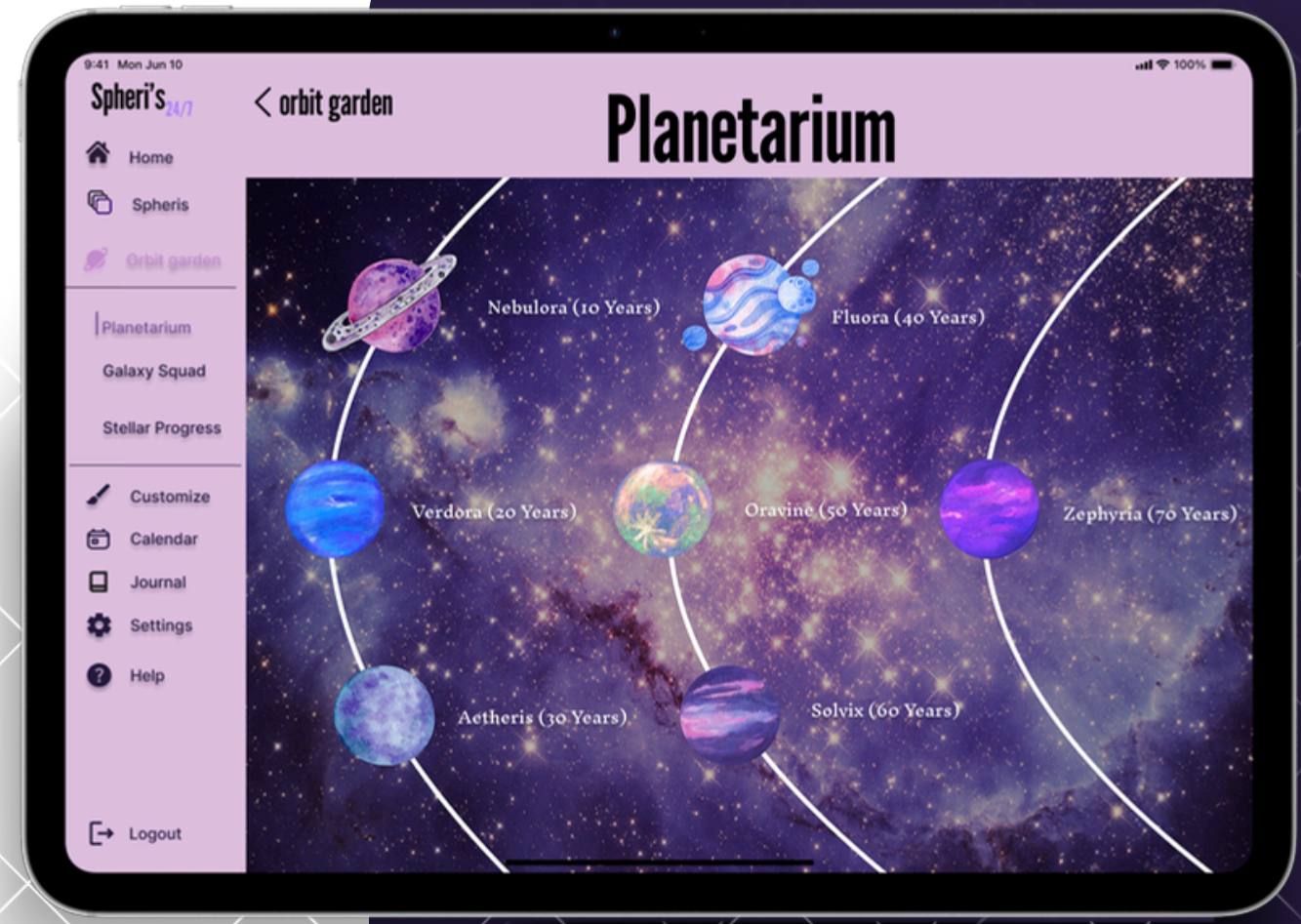




SPHERIS 24/7

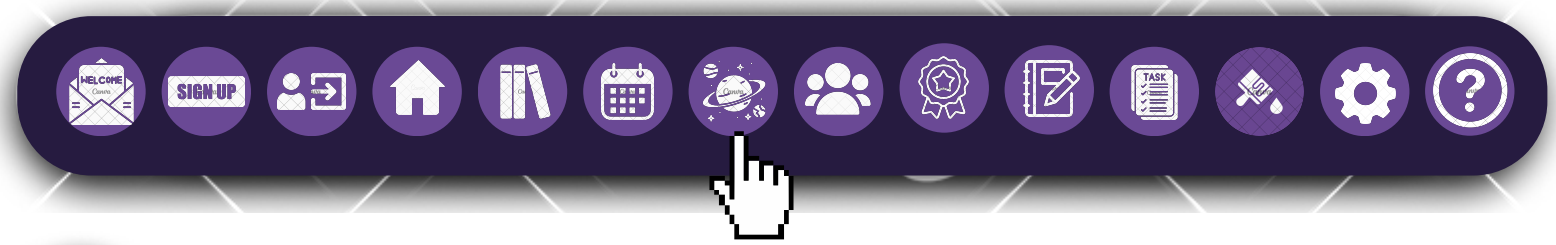
3.3 ON-SCREEN APPEARANCE REQUIREMENTS

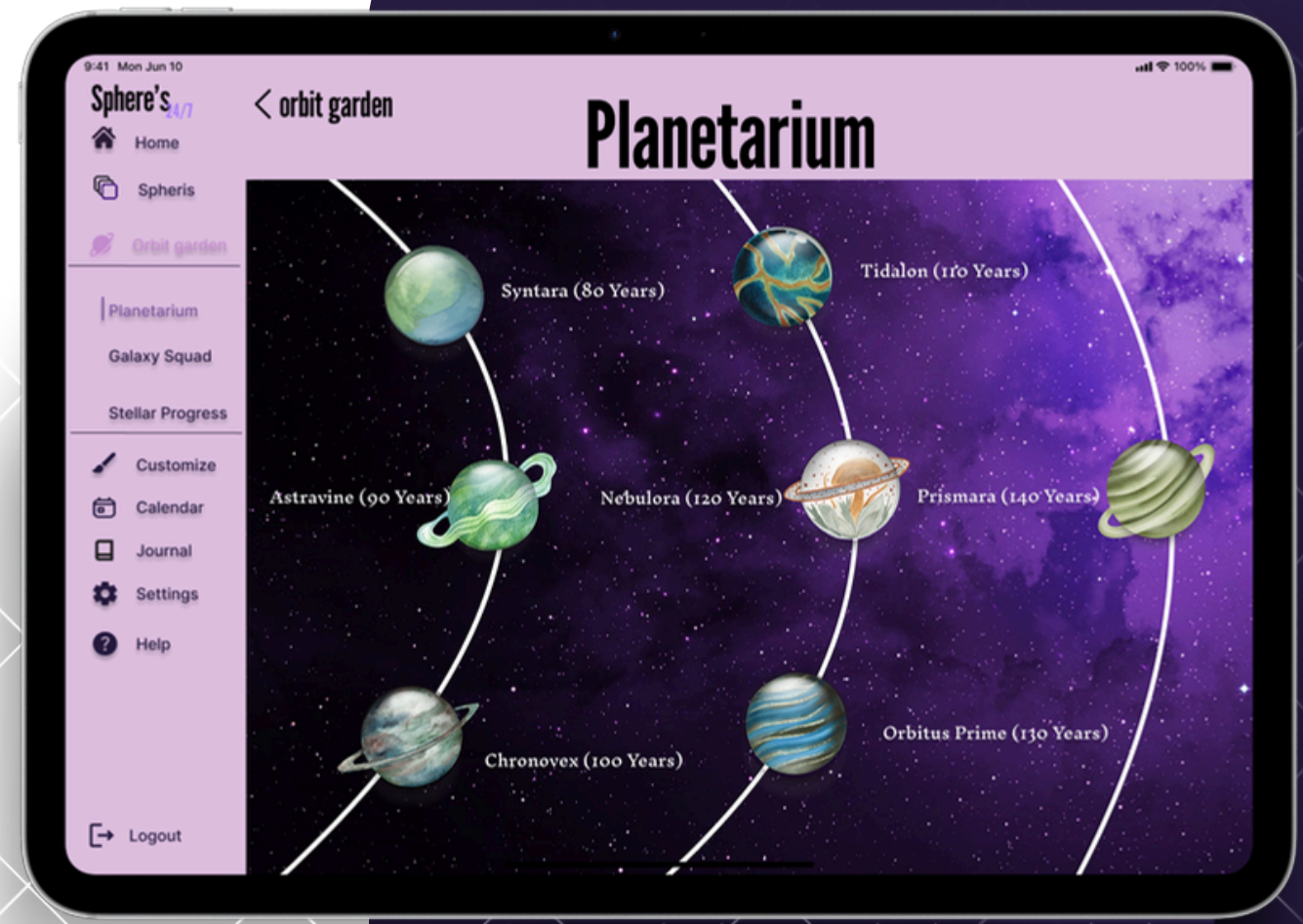




SPHERIS 24/7

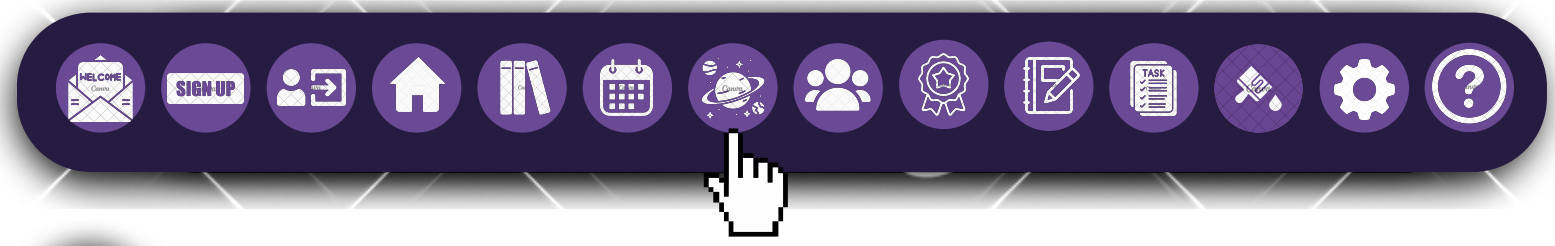
3.3 ON-SCREEN APPEARANCE REQUIREMENTS

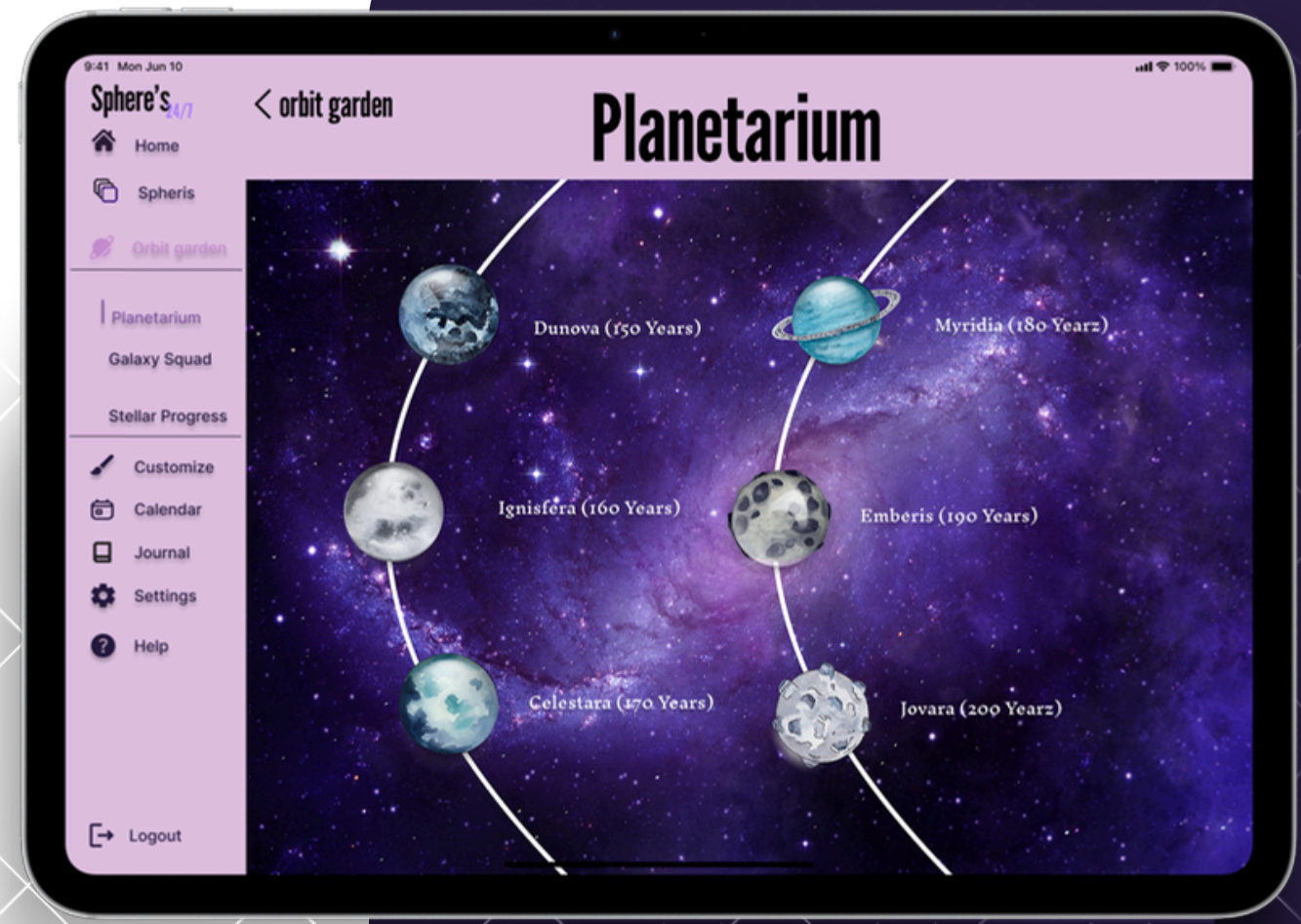




SPHERIS 24/7

3.3 ON-SCREEN APPEARANCE REQUIREMENTS



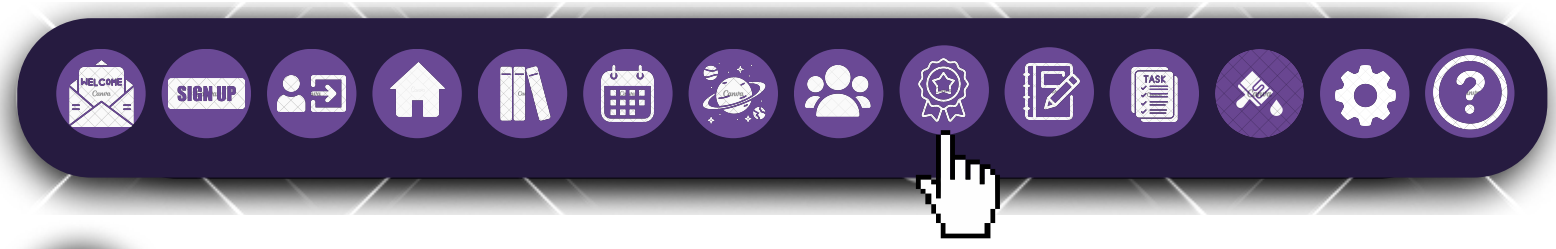


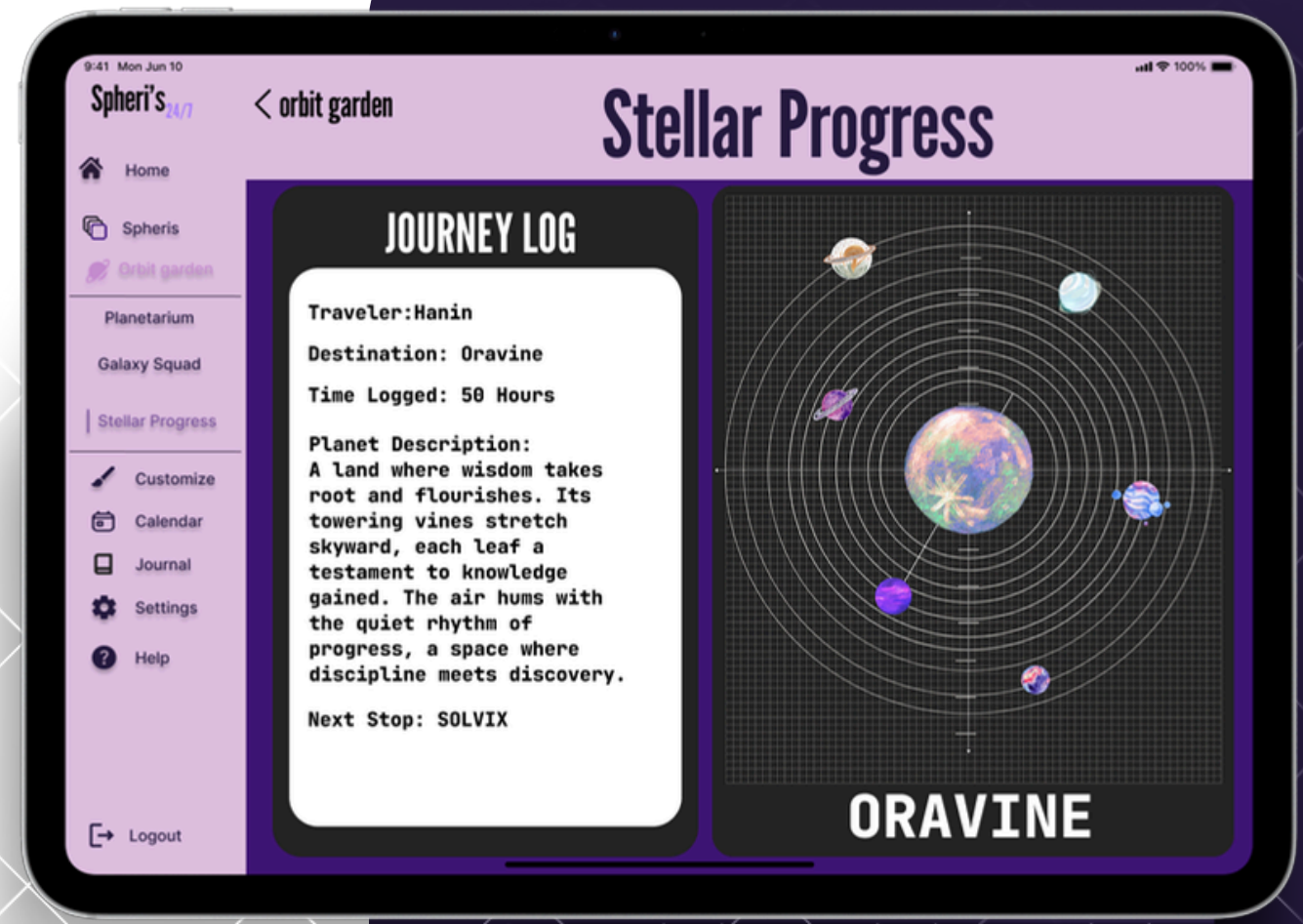
SPHERIS 24/7

3.3 ON-SCREEN APPEARANCE REQUIREMENTS

SPHERIS 24/7

3.3 ON-SCREEN APPEARANCE REQUIREMENTS



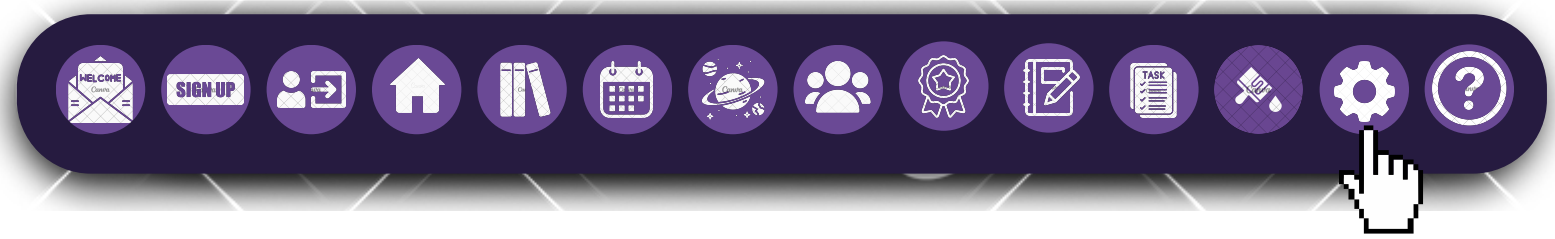


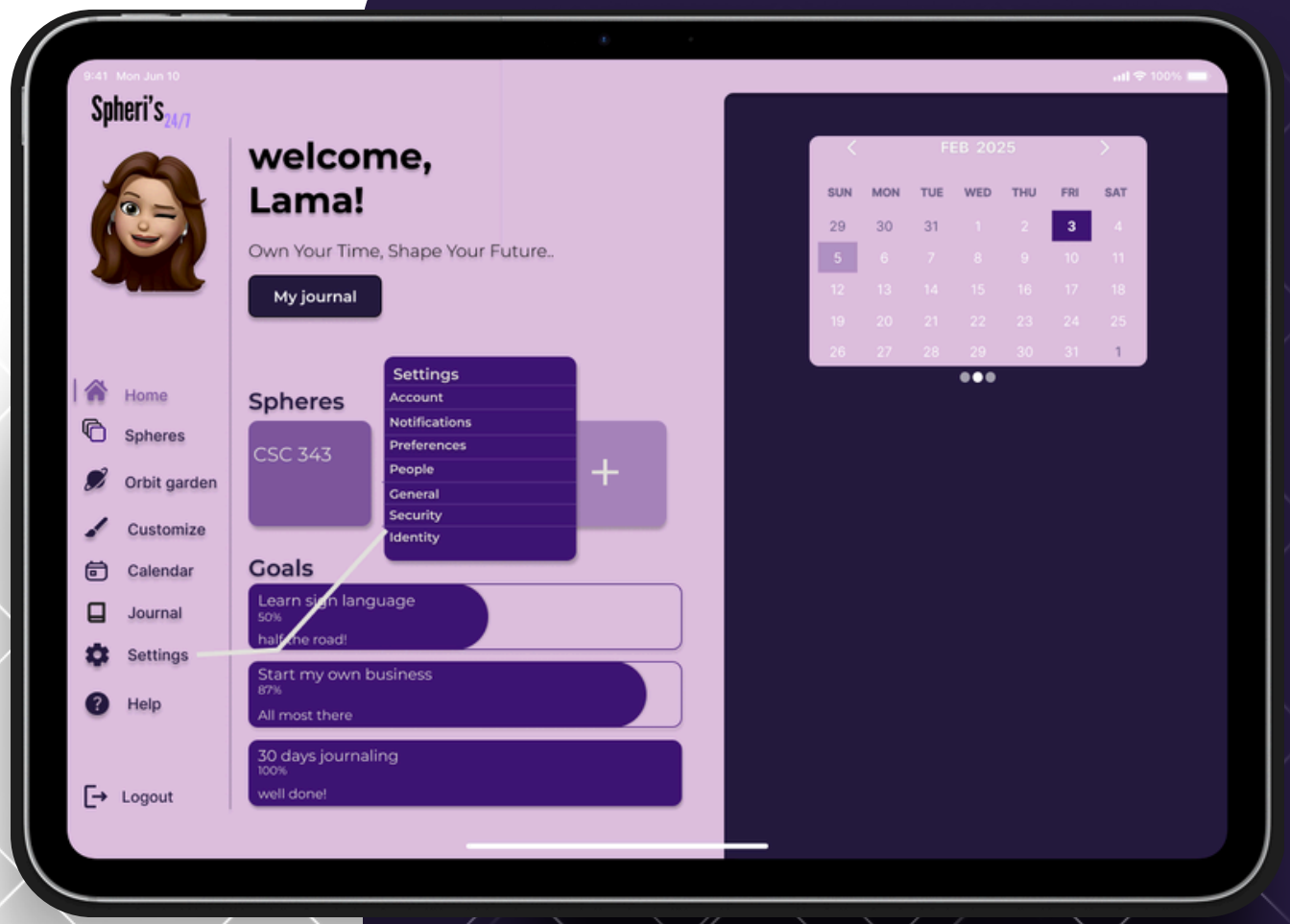
SPHERIS 24/7

3.3 ON-SCREEN APPEARANCE REQUIREMENTS

SPHERIS 24/7

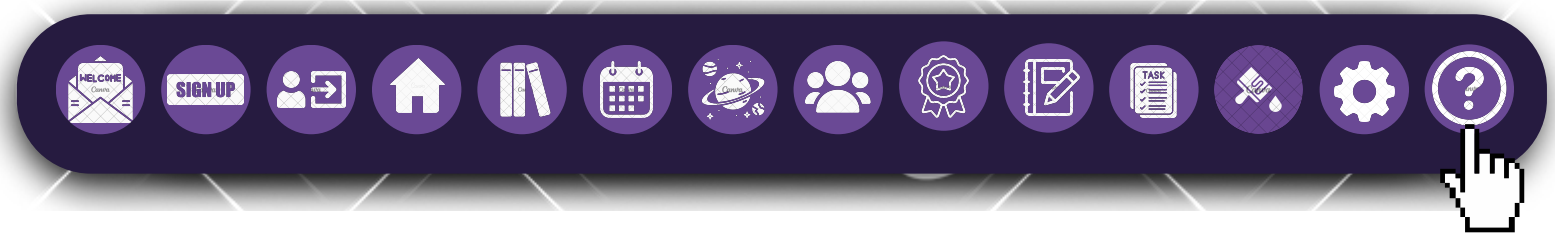
3.3 ON-SCREEN APPEARANCE REQUIREMENTS

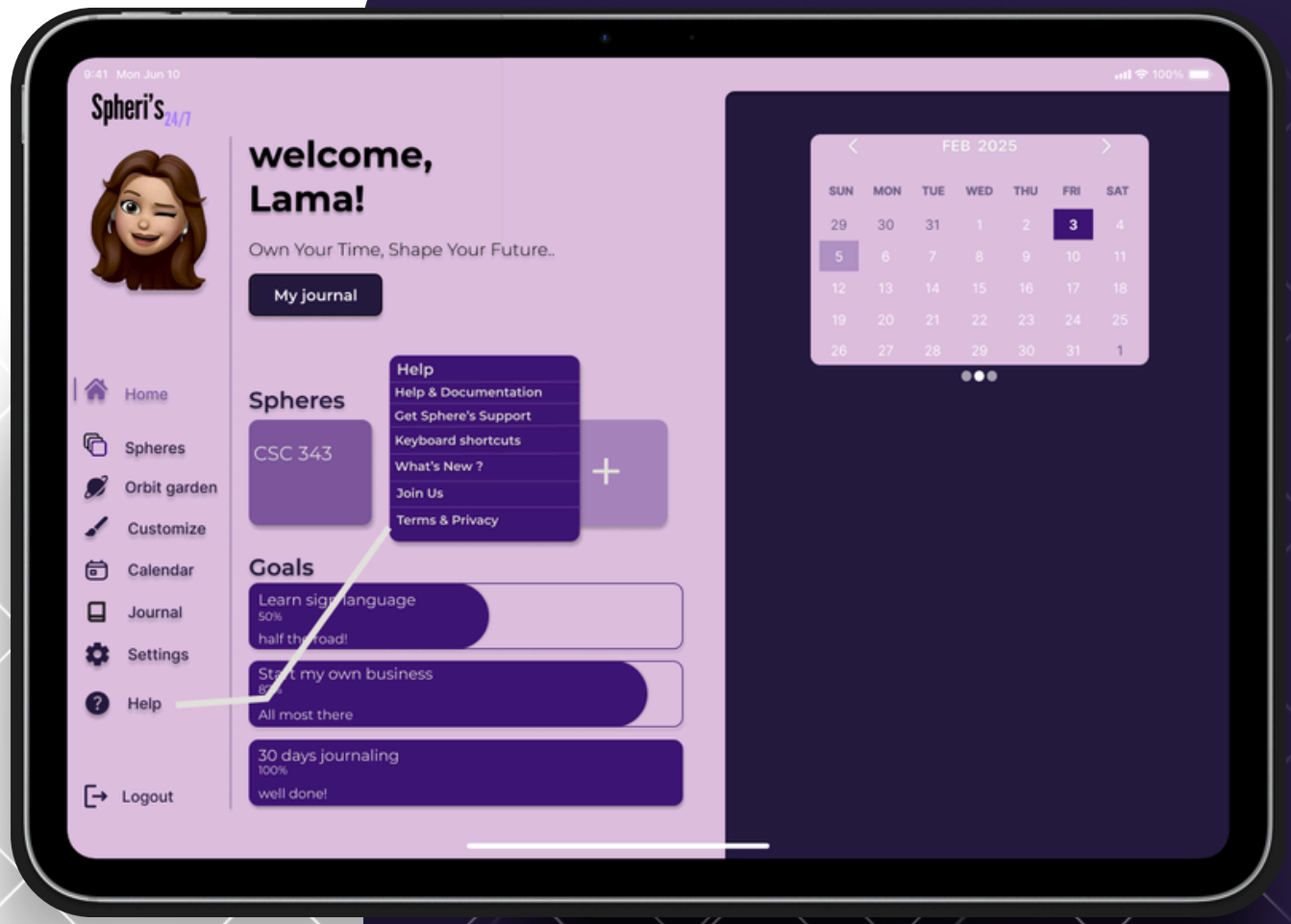




SPHERIS 24/7

3.3 ON-SCREEN APPEARANCE REQUIREMENTS





SPHERIS 24/7

04. FUNCTIONAL REQUIREMENTS SPECIFICATION

4.1 STAKEHOLDERS

Students & Academics	To manage assignments, deadlines, and study schedules efficiently.
Professionals & Entrepreneurs	To optimize productivity, track projects, and stay on top of deadlines.
Teams & Project Managers	To streamline task delegation, progress tracking, and collaboration.
Neurodivergent Individuals (e.g., ADHD)	To enhance focus and organization through structured planning and reminders.
Self-Improvement Enthusiasts	To set and achieve personal goals in fitness, learning, and skill development.

4.2 ACTORS AND GOALS

Actor	Type	Goal
User	Initiating	People who use the application to manage their lives, goals, tasks and schedule
Database	Participating	Device to store the user's data and information
System administrator	Participating	Manages the system updates and security
Email	Participating	system to send notifications and reminders

4.3 USE CASE CASUAL DESCRIPTION

SPHERIS 24/7

USE CASE ID	NAME	SHORT DESCRIPTION	CORRESPONDING REQ-ID
UC1	Create Account	Allows user to create its Account by Entering their Information	REQ1
UC2	Delete account	Allows user to delete its account	REQ31
UC3	Edit account	Allow user to Edit it's account	REQ30
UC4	Login	Allows user to sign in to it's Account by Entering their Credentials	REQ2
UC5	Logout	Allows user to Log out of their Account	REQ3
UC6	Customize View	Allows user to Customize their view based on their preferences to suit their needs	REQ4
UC7	initiate journal	Allows user to create their journal	REQ5
UC8	Add content to Journal	Allows user to write Contents into a Personal Journal	REQ6
UC9	Edit Journal	Allows user to Modify previously written journal entries	REQ7

4.3 USE CASE CASUAL DESCRIPTION

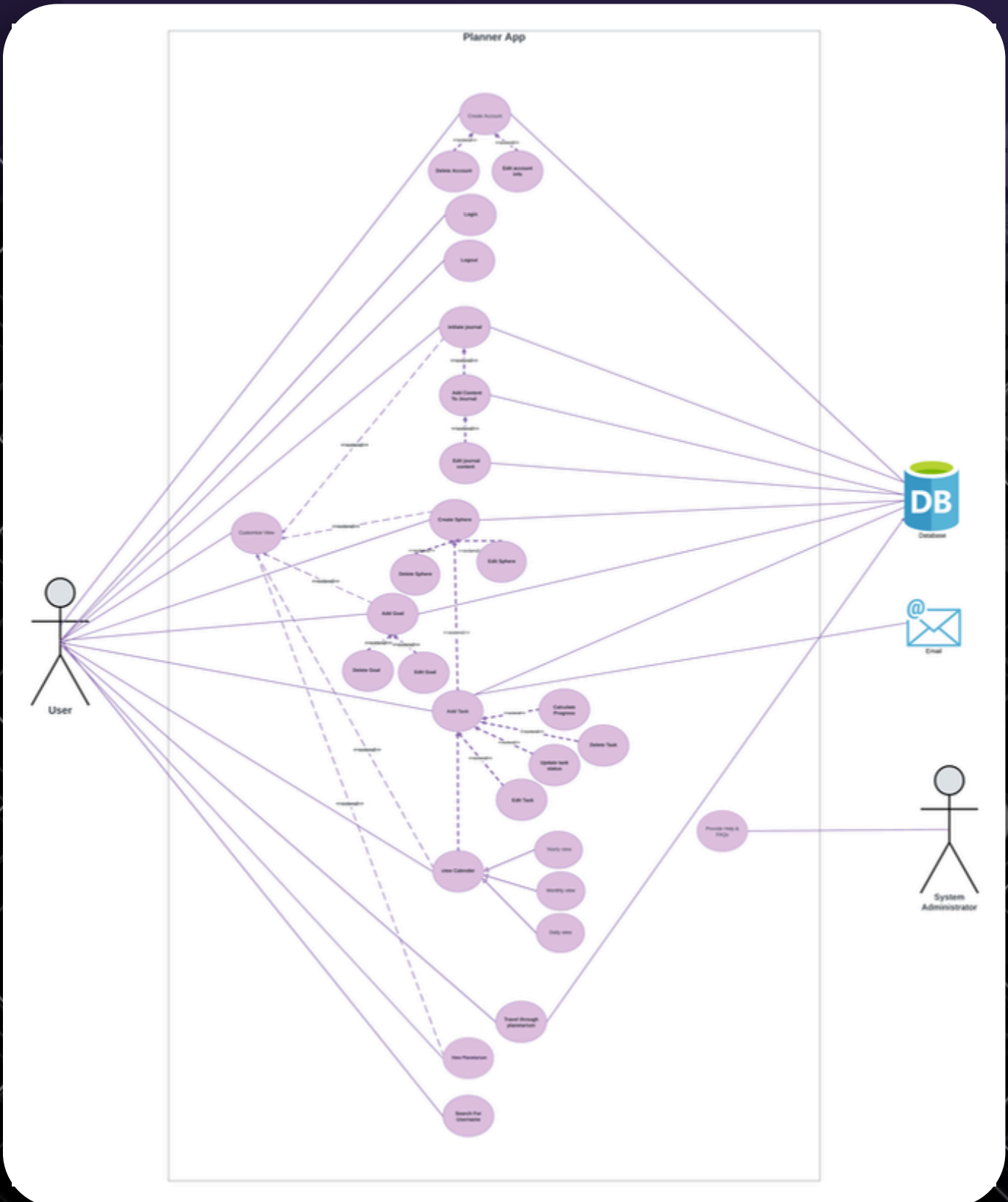
SPHERIS 24/7

use case id	name	short description	corresponding req-id
UC10	Create Spheris	Allows the user to Add a new spheris to their Environment	REQ8
UC11	Edit Spheris	Allows user to edit its spheris	REQ9
UC12	Delete Spheris	Allows the user to remove a Spheris from their Account	REQ10
UC13	Add Task	Allows the user to add a new task within a sphere, and sends email for special restrictions	REQ14,REQ17,REQ21
UC14	Edit Task	Allows the user to modify an existing task(such as changing details or deadlines)	REQ15
UC15	Delete Task	Allows the user to delete a task	REQ19
UC16	Update Task Status	Allow the user to update a task's status and progress as it changes	REQ20
UC17	Calculated Progress	Allows the user to View it's Progress	REQ18
UC18	Add Goal	Allows the user to add a goal	REQ11

4.3 USE CASE CASUAL DESCRIPTION

use case id	name	short description	corresponding req-id
UC19	Edit Goal	Allows users to edit it goal	REQ12
UC20	Delete Goal	Allow users to delete their goal	REQ13
UC21	View Calendar	Allows the user to view the calendar associated with spheris categories and tasks	REQ22
UC22	Yearly View	Allows user to view the calendar on a yearly basis	REQ21
UC23	Monthly View	Allows users to view the calendar on a monthly basis	REQ21
UC24	Daily View	Allows users to view the calendar on a daily basis	REQ21
UC25	Search for User	Allows the user to Search for another user	REQ26
UC26	Travel through planetarium	Allows user to travel from planet to another depending on the working hours	REQ23
UC27	View Planetarium	Allows the user to view all of the planetarium with its component (planet, planet description, galaxy squad, stellar journal)	REQ24,REQ25,REQ27,REQ28,REQ29
UC28	Provide Help & FAQs	Allows the System to provides help and FAQs to assist users with common issues	REQ32

4.4
USE CASE DIAGRAM



SPHERIS 24/7

4.5 USE CASE FULLY-DRESSED DESCRIPTION

SPHERIS 24/7

use case ID	UC6
Use case name	Customize View
Actor	user
Description	This use case describes the scenario where the user can customize it's own view based on their preferences to suit their needs ensuring flexibility
Stackeholders & Interests	Users who prefer controlling their view, aesthetic focused
Trigger	User desire a customizable and easy to use view

pre-condition	post-condition
1.The user shall be signed in 2.The user shall have access to customize	1.Customization is applied 2.Customization is saved

Normal flow
1.The user navigates to customization settings 2.The system displays customization tools 3.The user selects tool to add it in the view 4.The system display the tool in the view 5.The tool is saved in the view 6.{use case ends}
Alternative flow(extensions)
1.The user navigates to customization settings 2.The user exits the settings without saving the customizes tool 3.System wouldn't add the tool to the view 4.{use case ends}

4.5 USE CASE FULLY-DRESSED DESCRIPTION

SPHERIS 24/7

use case ID	UC8
Use case name	Add Content to journal
Actor	user
Description	This use case describes the scenario where the user can add content to its own journal
Stackeholders & Interests	Users who enjoy document their life
Trigger	User desire journaling

pre-condition	post-condition
1.The user shall be signed in 2.The user shall have access to the journal 3-the user shall create his journal	1.Content is added 2.Content is saved

Normal flow
1.The user navigates to journaling section 2.The System display the journal 3.The user adds content to the journal 4.The system receives the content 5.The content is saved in the journal 6.{use case ends}
Alternative flow(extensions)
1.The user navigates to journaling section 2.The System display the journal 3.the user attach an unreachable file 4.System would reject 5.{use case ends}

4.5 USE CASE FULLY-DRESSED DESCRIPTION

SPHERIS 24/7

use case ID	UC13
Use case name	Add task
Actor	user
Description	This use case describes the scenario where the user can create a new task to their spheris keeping tasks manageable
Stackeholders & Interests	users who prefer to keep track of their work
Trigger	Users need to list their tasks

pre-condition	post-condition
1.The user shall be signed in 2.The user shall create a sphere	1.Tasks are listed 2.Tasks are saved to spheris 3.timeline is initiated

Normal flow
1.The user navigates to sphere section 2.The System display the sphere view 3.The user add task 4.The system confirm task adding 5.The tasks are saves in the sphere 6.{use case ends}
Alternative flow(extensions)
1.The user navigates to sphere section 2.The System display the sphere view 3.The user add task with passed deadline 4.System would reject 5.{use case ends}

4.5 USE CASE FULLY-DRESSED DESCRIPTION

SPHERIS 24/7

use case ID	UC26
Use case name	Travel through planetarium
Actor	user
Description	This use case describes the scenario where the user can travel through planetarium by increasing the number of working hours
Stackeholders & Interests	users who work with motivation
Trigger	The user wants upgrade their plant

pre-condition	post-condition
1.The user shall be signed in 2.The user shall have access to planetarium	1.the system updates the plant

Normal flow
1.The user navigates to planetarium 2.The system retrieves and displays the planetarium 4.the user shall work for a selected time 5. The system shall update the users plant 6.{use case ends}
Alternative flow(extensions)
1.The user navigates to planetarium 2.The system retrieves and displays the planetarium 3.the user didn't work for the selected time 4.System would reject 5.{use case ends}

4.5 USE CASE FULLY-DRESSED DESCRIPTION

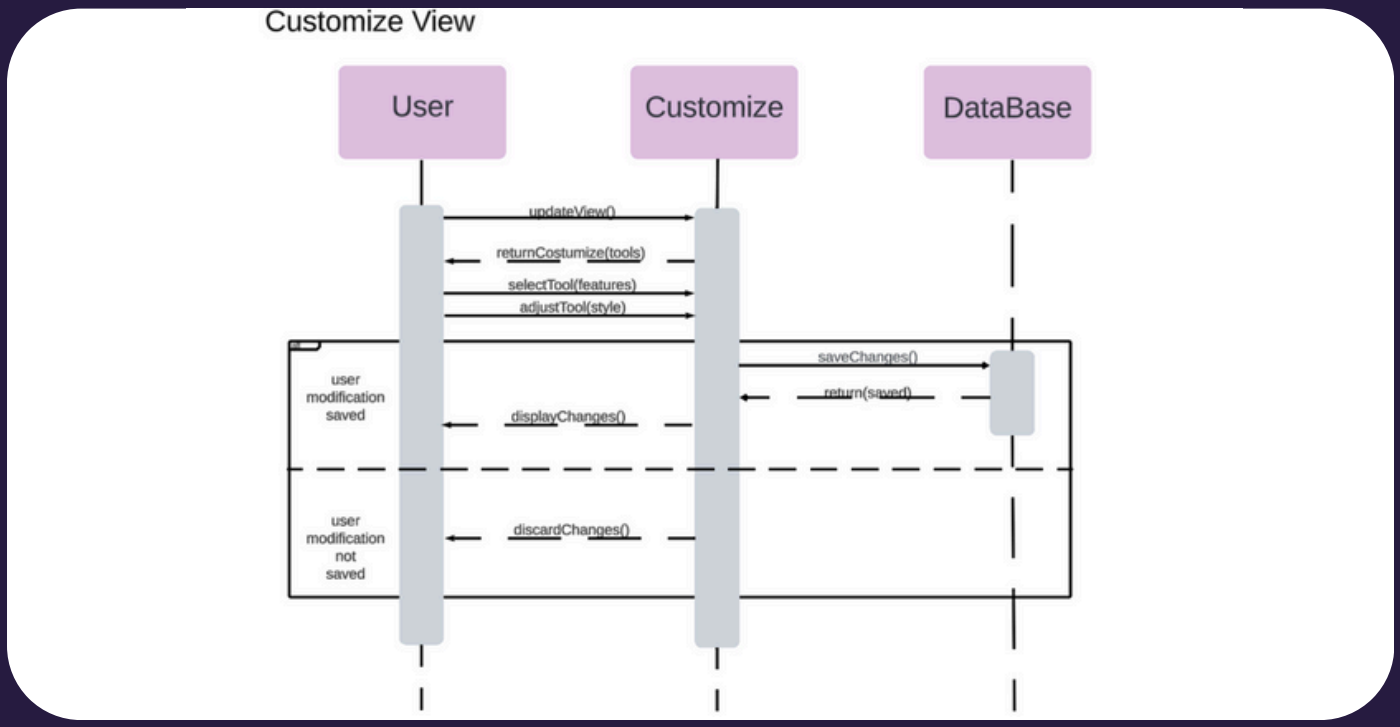
SPHERIS 24/7

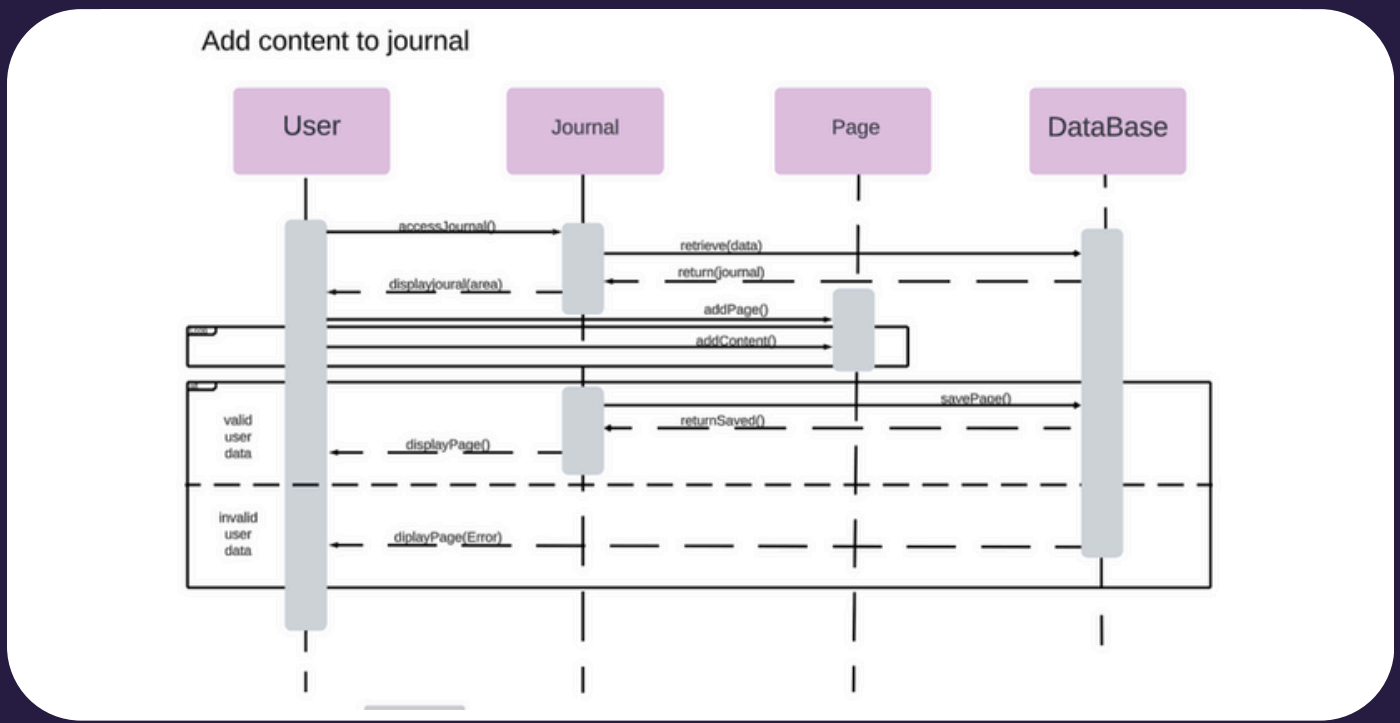
use case ID	UC21
Use case name	View Calendar
Actor	user
Description	This use case describes the scenario where the user can view the calendar with categorized tasks and manage them by adding or deleting tasks.
Stackeholders & Interests	organized users
Trigger	user need to schedule their tasks

pre-condition	post-condition
1.The user shall be signed in	1.calendar displayed

Normal flow
1.The user navigates to calendar 2.The system retrieves and displays the calendar 3.the user select a filtering view option 4.the system categories the tasks 5.the system display the calendar 6.{use case ends}
Alternative flow(extensions)
1.The user navigates to calendar 2.The system retrieves and displays the calendar 3.The user add task in not matched category 4.The system will display incorrect filtetring 5.{use case ends}

5.1
INTERACTION DIAGRAM
SEQUENCE DIAGRAM

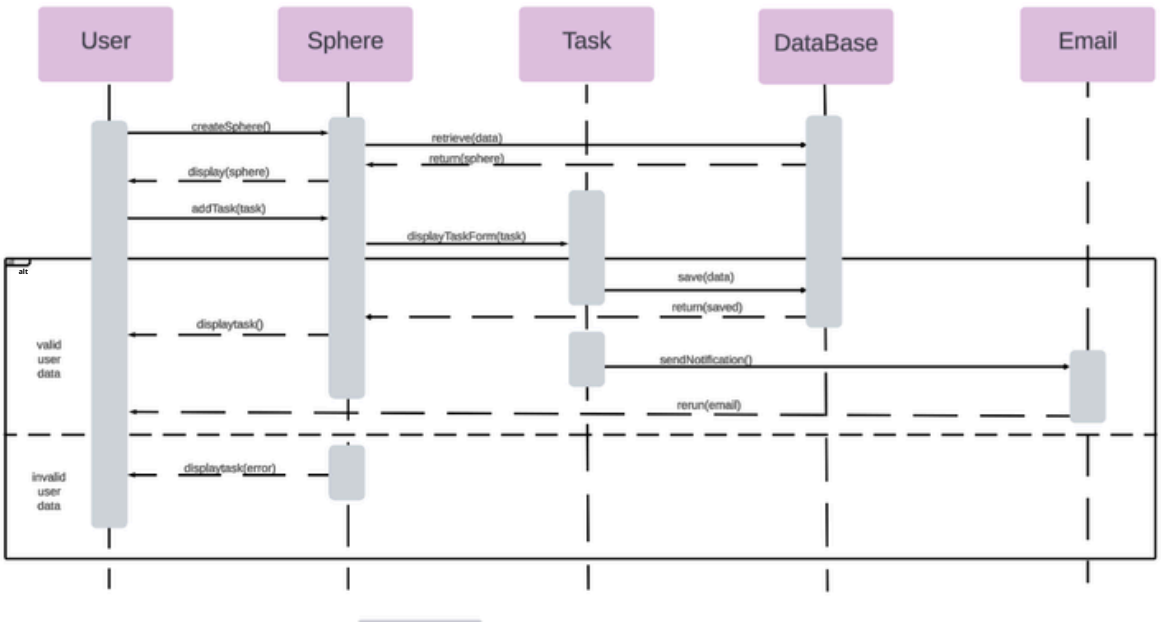




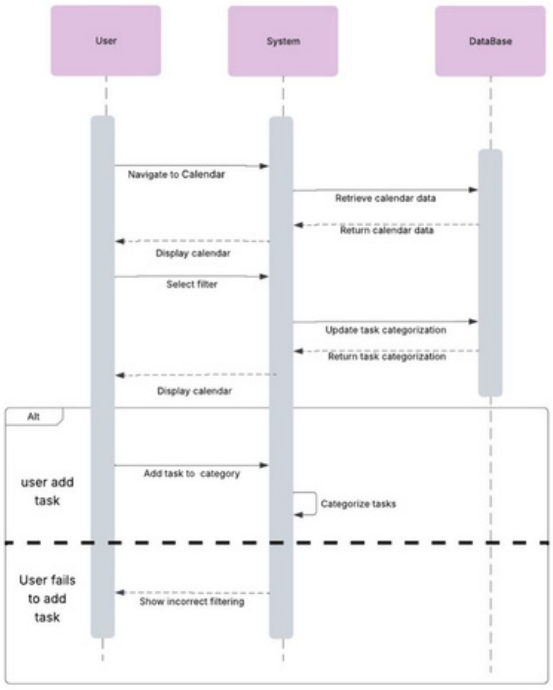
SPHERIS 24/7

5.1
INTERACTION DIAGRAM
SEQUENCE DIAGRAM

Add Task



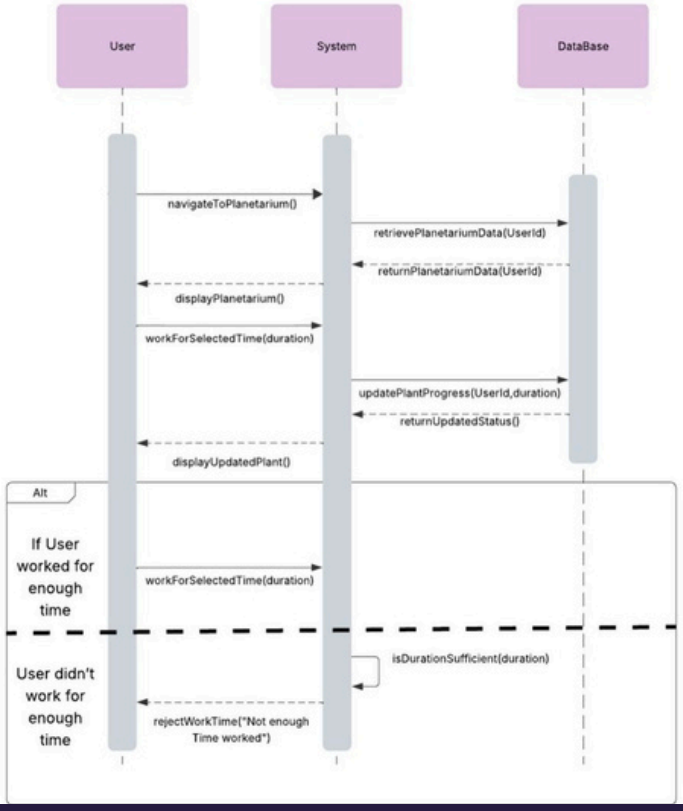
View Calendar



SPHERIS 24/7

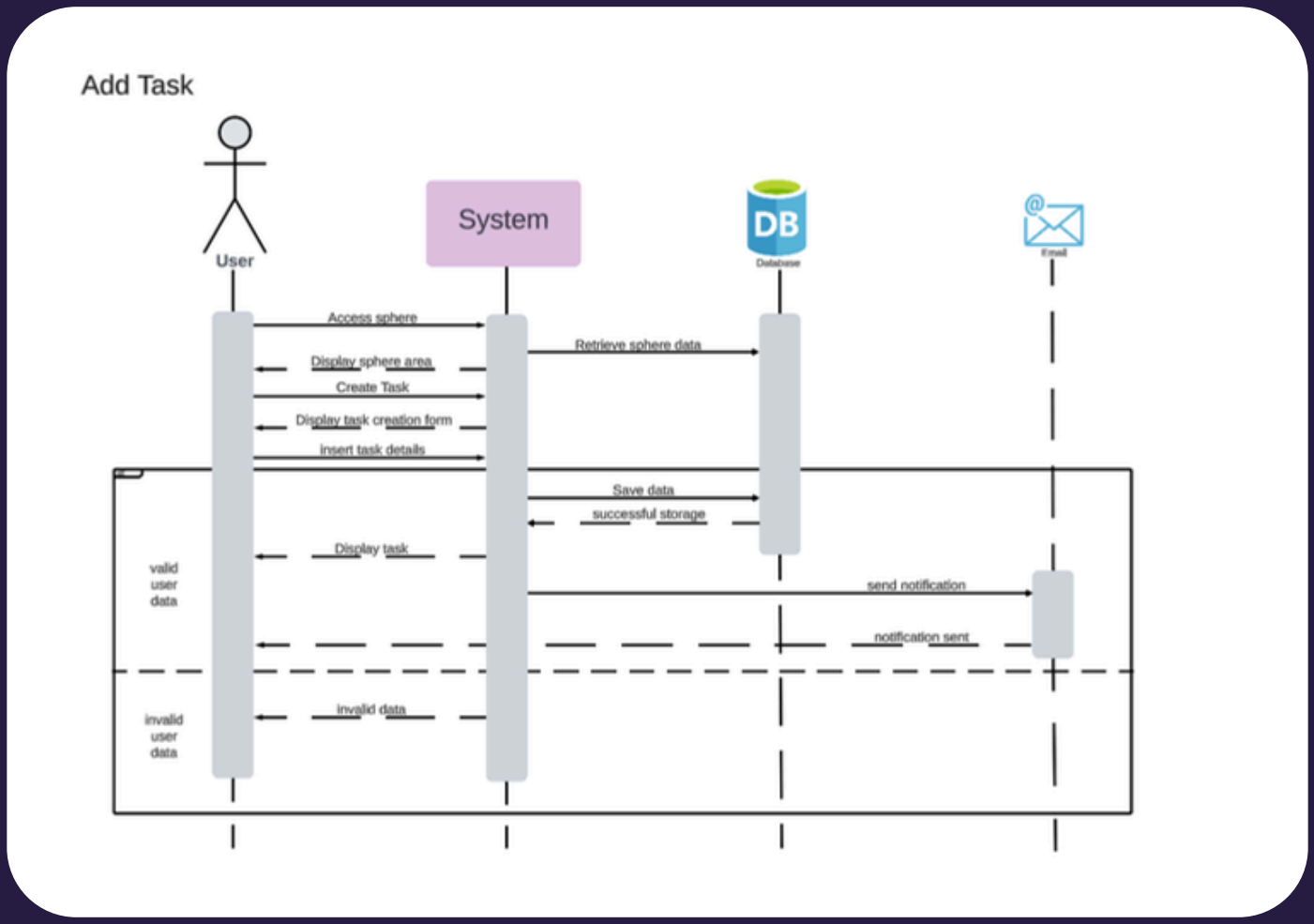
5.1
INTERACTION DIAGRAM
SEQUENCE DIAGRAM

Travel through planetarium



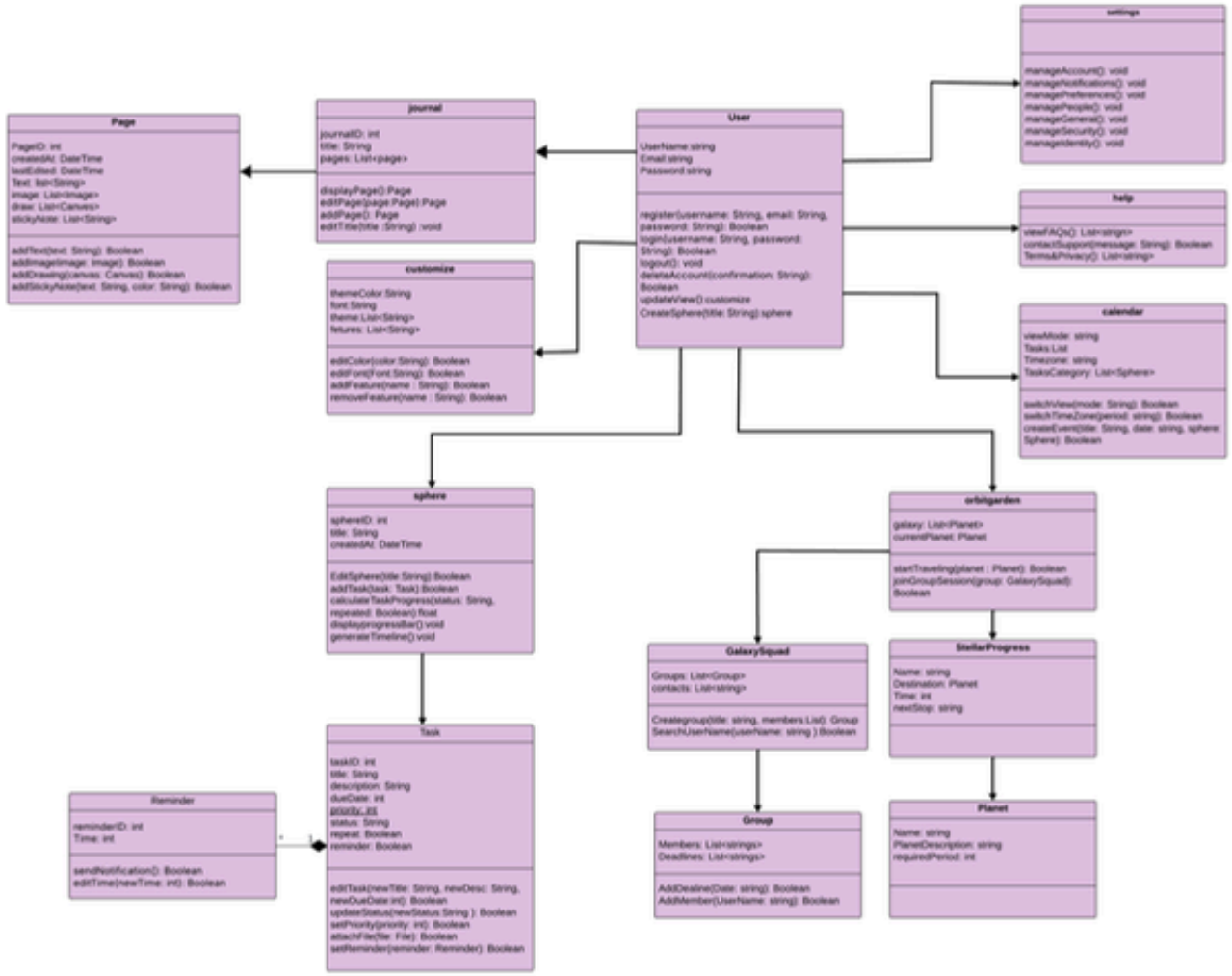
SPHERIS 24/7

5.2
INTERACTION DIAGRAM
SEQUENCE DIAGRAM



SPHERIS 24/7

5.3
CLASS DIAGRAM



SPHERIS 24/7

Class diagram description

User class

ATTRIBUTES

UserName	type: string Description: The username associated with the user's account .
Email	type: string Description: The email address associated with the user, which is used for communication and sending notifications or account-related information.
Password	type: string Description: The password associated with the user's account, which is used for authenticating the user when they log into the system.

METHODS

register(username: String, email: String, password: String)	Return Type: Boolean Description: This method allows a new user to register by providing their username, email, and password, and returns true if the registration is successful, and false if there is an error.
login(username: String, password: String):	Return Type: Boolean Description: This method allows an existing user to log in using their username and password, and returns true if the login is successful and false if the credentials are incorrect.
logout():	Return Type: void Description: This method allows the user to log out of the system, terminating their session.
deleteAccount(confirmation: String):	Return Type: Boolean Description: This method allows a user to delete their account after confirming their action. It returns true if the account is successfully deleted, and false if the deletion fails.
updateView():	Return Type: customize Description: This method allows the user to update their view preferences within the system.
createSphere(title: String):	Return Type: sphere Description: This method allows the user to create a new sphere (a specific category or section for organizing tasks or information).

Class diagram description

Journal class

ATTRIBUTES

journalID	Type: int Description: A unique identifier for the journal.
title	Type: String Description: The title of the journal.
pages	Type: List<Page> Description: A list of Page objects. Each page represents a part of the journal, where users can add content such as text, images, drawings, etc.

METHODS

displayPage()	Return Type: Page Description: Retrieves and returns a specific page from the journal.
editPage(page:Page)	Return Type: Page Description: Allows users to edit on page attributes
addPage():	Return Type: Page Description: Adds a new Page to the journal, and returns the newly added Page object, allowing users to organize content effectively within the journal.
editTitle(title:String)	Return Type: void Description: Allows users to change the title of its journal.

page class

PageID	Type: int Description: This is a unique identifier for each page, which helps the system distinguish one page from another.
createdAt	Type: DateTime Description: This shows the date and time when the page was created, helping users or the system track when the content was added.

Class diagram description
page class

ATTRIBUTES

lastEdited	Type: DateTime Description: This timestamp shows when the page was last edited, which helps keep track of changes made to the page.
Text	Type: List<String> Description: This stores the text content of the page, including paragraphs, headings, or any other written content the user adds.
images	Type: List<Image> Description: This is a list of images attached to the page, Users can insert pictures to make the page more visually appealing or informative.
draw	Type: List<Canvas> Description: This is where any drawings or diagrams made by the user will be stored. These could be sketches, charts, or any other custom visual content.
stickyNote	Type: List<String> Description: Users can add sticky notes with text to the page, which could be quick reminders, ideas, or important notes

METHODS

addText(text: String):	Return Type: Boolean This method allows users to add text to the page. It returns true if the text was added successfully.
addImage(image: Image):	Return Type: Boolean This lets users add an image to the page. The system returns true if the image is successfully uploaded.
addDrawing(canvas: Canvas):	Return Type: Boolean This allows users to add a drawing (like a chart or diagram) to the page. It returns true if the drawing is added successfully.
addStickyNote(text: String, color: String):	Return Type: Boolean This method allows users to add a sticky note to the page, where they can enter text and choose a color for the note.

Class diagram description

Sphere class

ATTRIBUTES

sphereID	Type: int Description: A unique identifier for each sphere, ensures that each sphere can be distinguished from others in the system.
title	Type: String Description: The title of the sphere. This helps users easily identify the purpose of the sphere.
createdAt	Type: DateTime Description: The date and time when the sphere was created, which helps track when the sphere was initially created and is useful for historical data.

METHODS

EditSphere(title: String)	Return Type: Boolean Description: This method allows users to edit the title of the sphere., and returns true if the title was successfully updated, and false if an error occurred.
addTask(task: Task):	Return Type: Boolean Description: This method adds a task to the sphere, The task is passed as an argument, and the method returns true if the task was successfully added.
calculateTaskProgress(status: String, repeated: Boolean):	Return Type: float Description: This method calculates the progress of a task based on its status (e.g., "In Progress", "Completed") and whether the task is repeated or one time, and returns the percentage of progress as a float.
displayProgressBar():	Return Type: void Description: This method displays a progress bar to visually show the completion status of the tasks within the sphere.
generateTimeline():	Return Type: void Description: This method generates a timeline for the tasks in the sphere, allowing users to see the task deadlines or milestones.

Class diagram description

Task class

ATTRIBUTES

taskID	Type: int Description: A unique identifier for each task, ensures that each task can be easily referenced within the system.
title	Type: String Description: The title of the task. This helps the user quickly identify what the task is about.
description	Type: String Description: A detailed description of the task. It provides more information on what needs to be done for the task.
dueDate	Type: int Description: The date by which the task must be completed, which helps prioritize tasks based on deadlines.
Priority	Type: int Description: The priority level of the task (e.g., low, medium, high). This allows users to sort and focus on more important tasks.
status	Type: String Description: The current status of the task, such as "Pending", "In Progress", or "Completed", which helps users track the progress of the task.
repeat	Type: Boolean Description: A flag indicating whether the task repeats on a regular basis (e.g., daily, weekly). If true, the task will recur; if false, the task is one-time only.
reminder	Type: Boolean Description: A flag indicating whether a reminder is set for the task. If true, the system will notify the user about the task at the specified time.

METHODS

editTask(newTitle: String, newDesc: String, newDueDate: int):	Return Type: Boolean Description: This method allows users to edit the title, description, and due date of an existing task. It returns true if the task was successfully updated, and false otherwise.

Class diagram description

Task class

METHODS

updateStatus(newStatus: String):	Return Type: Boolean Description: This method updates the status of the task (e.g., from "Pending" to "Completed"). It returns true if the status was updated successfully.
attachFile(file: File):	Return Type: Boolean Description: This method allows users to attach a file (e.g., document or image) to the task, returns true if the file was attached successfully.
setReminder(reminder: Reminder):	Return Type: Boolean Description: This method allows users to set a reminder for the task, returns true if the reminder was set successfully.

Reminder class

ATTRIBUTES

reminderID:	Type: int Description: A unique identifier for each reminder, which helps the system distinguish one reminder from another.
Time:	Type: int Description: Represents the time at which the reminder is set. This is stored as an integer (likely representing time in minutes or hours) to trigger the notification at the appropriate time.

METHODS

sendNotification():	Return Type: Boolean Description: This method sends a notification to the user when the reminder time arrives, returns true if the notification was successfully sent, and false if there was an error.
editTime(newTime: int):	Return Type: Boolean Description: This method allows users to change the reminder time. It takes a new time (in minutes or hours) as input and updates the reminder. It returns true if the time was successfully updated.

Class diagram description

calendar class

ATTRIBUTES

viewMode:	Type: string Description: Represents the current display mode of the calendar, such as "daily", "weekly", or "monthly." This determines how the calendar is presented to the user.
Tasks:	Type: List Description: A list of tasks associated with the calendar. This allows users to track and manage tasks directly from their calendar view.
Timezone:	Type: string Description: The time zone used by the calendar. This ensures that all events and tasks are displayed at the correct time, taking into account the user's local time zone.
TasksCategory:	Type: List<Sphere> Description: A list of task categories, where each category is represented by a Sphere. This helps organize tasks under specific groups or projects.

METHODS

switchView(mode: String):	Return Type: Boolean Description: This method allows the user to switch the view mode of the calendar (e.g., from daily to weekly or monthly). It returns true if the view was successfully switched.
switchTimeZone(period: string):	Return Type: Boolean Description: This method allows users to switch the calendar's time zone, adjusting the display for different time periods. It returns true if the time zone was successfully switched.
createEvent(title: String, date: string, sphere: Sphere):	Return Type: Boolean Description: This method enables users to create a new event on the calendar. The event includes a title, date, and is associated with a specific Sphere (task category). It returns true if the event was successfully created.

Class diagram description

orbitgarden class

ATTRIBUTES

galaxy:	Type: List<Planet> Description: A list of planets within the galaxy. This property represents all the planets available for exploration within the galaxy.
currentPlanet:	Type: Planet Description: Represents the current planet that the user is on. It provides the user's current location within the galaxy.

METHODS

startTraveling(planet: Planet):	Return Type: void Description: This method allows the user to start traveling to a new planet within the galaxy. It updates the current Planet attribute to the selected planet, effectively changing the user's location by decreasing time period of plant.
joinGroupSession(group: GalaxySquad):	Return Type: void Description: This method allows the user to join a group session with a specified GalaxySquad. It provides the user with the ability to collaborate with other users in a shared session, likely for tasks or exploration.

StellarProgress class

ATTRIBUTES

Name:	Type: string Description: The name of the user in the mission.
Destination:	Type: string Description: The destination planet the user is traveling to.
Time:	Type: int Description: The time remaining for the journey. This could be represented as an integer (e.g., in minutes, hours, or days), indicating how long until the user reaches their destination.
nextStop:	Type: string Description: The next stop planet on the journey.

Class diagram description

Planet class

ATTRIBUTES

Name:	Type: string Description: The name of the planet, providing the identifier for the planet, allowing users to recognize and refer to different planets within the system.
PlanetDescription:	Type: string Description: A brief description of the planet, including details about the planet's characteristics, environment, or significance within the system.
requiredPeriod:	Type: int Description: The number of hours required to complete tasks or activities on the planet, which represent the time needed for a user to fully engage with tasks on that planet, or the time required for a journey.

GalaxySquad class

ATTRIBUTES

Groups:	List<Group> Description: A list of groups that belong to the galaxy squad, each group may consist of multiple members and represent a specific project or team within the galaxy.
contacts:	Type: List<string> Description: A list of contacts within the GalaxySquad, containing usernames of available contacts.

METHODS

CreateGroup(title: string, members: List):	Return Type: group Description: This method allows users to create a new group by specifying a title and a list of members, returning newly created Group.
SearchUserName(userName : string):	Return Type: Boolean Description: This method searches for a user by their username within the GalaxySquad system, returns true if the user is found, and false if the user does not exist.

Class diagram description

Group class

ATTRIBUTES

Members:	Type: List<string> Description: A list containing the usernames of all the members in the group. This keeps track of who's in the group and is useful for organizing group-related activities and tasks.
Deadlines:	Type: List<string> Description: A list representing the deadlines shared among group members.

METHODS

AddDeadline(Date: string):	Returns: Boolean Description: This method allows the group to add a new deadline. The date is provided as a string, and it returns true if the deadline is successfully added, or false if there's an issue.
AddMember(UserName: string):	Returns: Boolean Description: This method adds a new member to the group by taking a username as input. If the member is successfully added, it returns true; if there's an issue, it returns false.

settings class

METHODS

manageAccount():	Return Type: void Description: Opens the settings section where the user can update account-related information such as username, email, or password.
manageNotifications():	Return Type: void Description: Allows the user to control how and when they receive notifications.
managePreferences():	Return Type: void Description: This method enables the user to adjust their system preferences, such as interface settings, language, and theme customizations.

Class diagram description

settings class

METHODS

managePeople():	Return Type: void Description: Manages user contacts, blocklist, and any shared or followed users.
manageGeneral():	Return Type: void Description: A general management function for other settings not covered by the other categories. This may include privacy settings, data management, or system-wide configuration.
manageSecurity():	Return Type: void Description: This method allows the user to adjust security-related settings, such as changing passwords, enabling two-factor authentication, or managing login sessions.
manageIdentity():	Return Type: void Description: Allows the user to edit their public display profile or avatar.

Help class

METHODS

viewFAQs():	eturn Type: List<string> Description: Returns a list of frequently asked questions (FAQs). This method allows users to view the most common queries and answers related to the system.
contactSupport(message: String):	Return Type: Boolean Description: Sends a support message to the system's support team. The method returns true if the message was successfully sent and false if there was an issue.
Terms&Privacy():	Return Type: List<string> Description: Returns a list of terms and privacy policies that users need to be aware of. This ensures users are informed about the system's rules and data handling practices.

Class diagram description
customize class

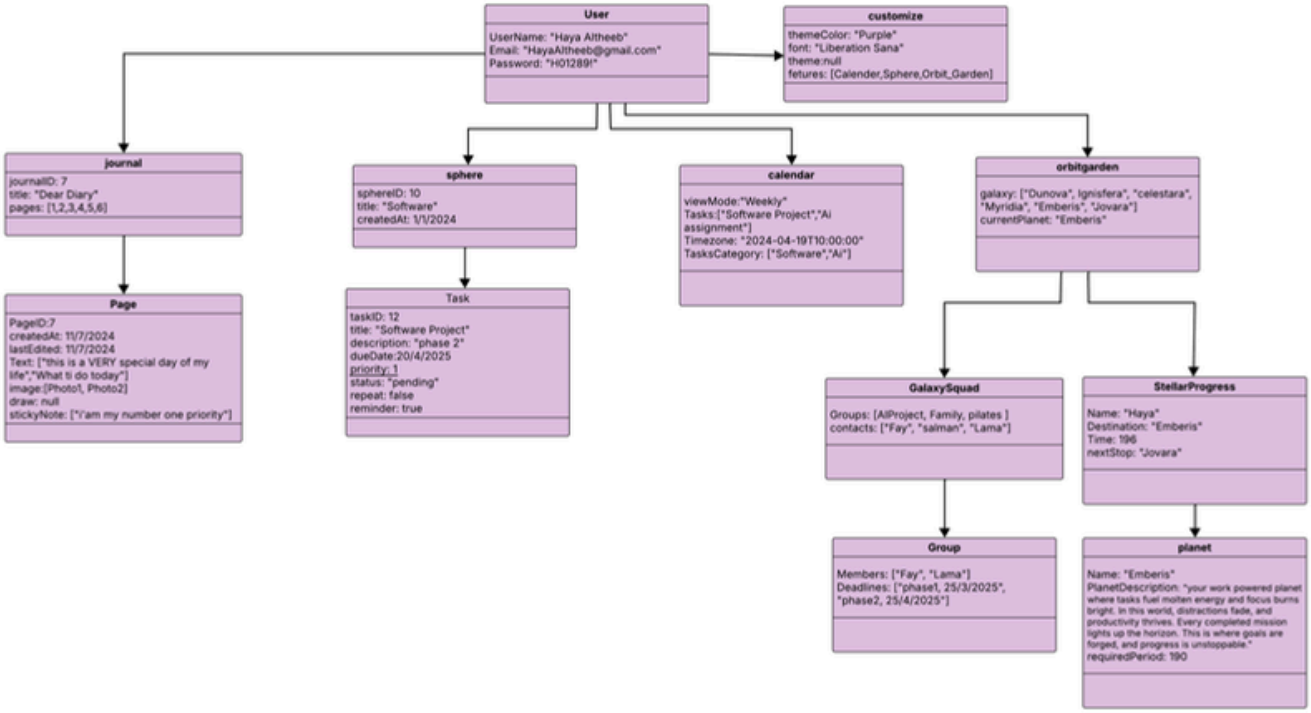
ATTRIBUTES

themeColor:	Type: String Description: Represents the color scheme of the interface. This allows users to set the visual theme of the platform according to their liking.
font:	Type: String Description: Represents the font style used in the platform. Users can customize the font to match their visual preferences or improve readability.
theme:	Type: List<String> Description: A list of predefined theme options available to the user. This allows users to choose from different themes or customize their own.
fetures:	Type: List<String> Description: A list of strings representing the active features/tools (e.g., journal, calendar) displayed on the homepage.

METHODS

editColor(color: String):	Return Type: Boolean Description: This method allows users to change the theme color of the interface. It returns true if the color change was successfully applied.
editFont(font: String):	Return Type: Boolean Description: Allows the user to change the font style used across the system. It returns true if the font change was successful.
addFeature(name : String):	Return Type: Boolean Description: Adds a feature/tool (e.g., “Calendar”, “Journal”) to the homepage view.
removeFeature(name : String):	Return Type: Boolean Description: Removes the selected feature/tool from the homepage view.

DOMAIN MODELING



SPHERIS 24/7

5.4.1 Architectural diagram

Architectural style

IN SPHERIS 24/7 WE CHOSE A **CLIENT-SERVER ARCHITECTURE** AS THE FOUNDATION OF OUR STYLE, DUE TO ITS STRONG SUPPORT FOR SECURITY, SCALABILITY, AND MAINTAINABILITY-ALL OF WHICH ARE ESSENTIAL TO DELIVER A RELIABLE EXPERIENCE ACROSS OUR PLATFORM FEATURES LIKE JOURNALING, CALENDARS, SPHERES, TASKS AND THE ORBIT GARDEN, FOR EXAMPLE
THE CLIENT SENDS REQUESTS (LIKE SAVING A JOURNAL PAGE OR UPDATING A TASK), AND THE SERVER RESPONDS BY EXECUTING THE OPERATION AND RETURNING THE RESULT.

THIS ARCHITECTURE IS IDEAL FOR OUR PROJECT BECAUSE IT HELPS US MAINTAIN A CENTRALIZED SYSTEM, MEANING ALL USER DATA IS SAFELY STORED AND MANAGED IN ONE PLACE, WHICH MAKES IT MORE SECURE, SINCE SENSITIVE DATA ISN'T SCATTERED ACROSS DEVICES AND WE CAN IMPLEMENT STRONG SECURITY MEASURES LIKE AUTHENTICATION AND AUTHORIZATION DIRECTLY ON THE SERVER.
IT ALSO MAKES THE SYSTEM SCALABLE, SO IF MORE USERS JOIN, OR WE EXPAND FEATURES, WE CAN UPGRADE SERVER PERFORMANCE WITHOUT AFFECTING INDIVIDUAL CLIENT DEVICES, EACH CLIENT REMAINS LIGHTWEIGHT AND RESPONSIVE.
ADDITIONALLY IT'S EASIER TO MAINTAIN, BECAUSE WHEN WE UPDATE OR FIX SOMETHING (LIKE A BUG OR FEATURE), WE ONLY NEED TO MAKE CHANGES ON THE SERVER,AND THE CLIENT APPS DON'T NEED TO BE REINSTALLED OR MANUALLY UPDATED BY THE USER, EVERYTHING WORKS SEAMLESSLY ACROSS DEVICES.
FOR A SYSTEM LIKE OURS THAT INCLUDES MULTI-FUNCTIONAL FEATURES SUCH AS JOURNALING, TASK MANAGEMENT, SCHEDULING, AND GROUP INTERACTIONS, CLIENT-SERVER IS THE MOST PRACTICAL AND RELIABLE CHOICE

5.4.2 Architectural diagram

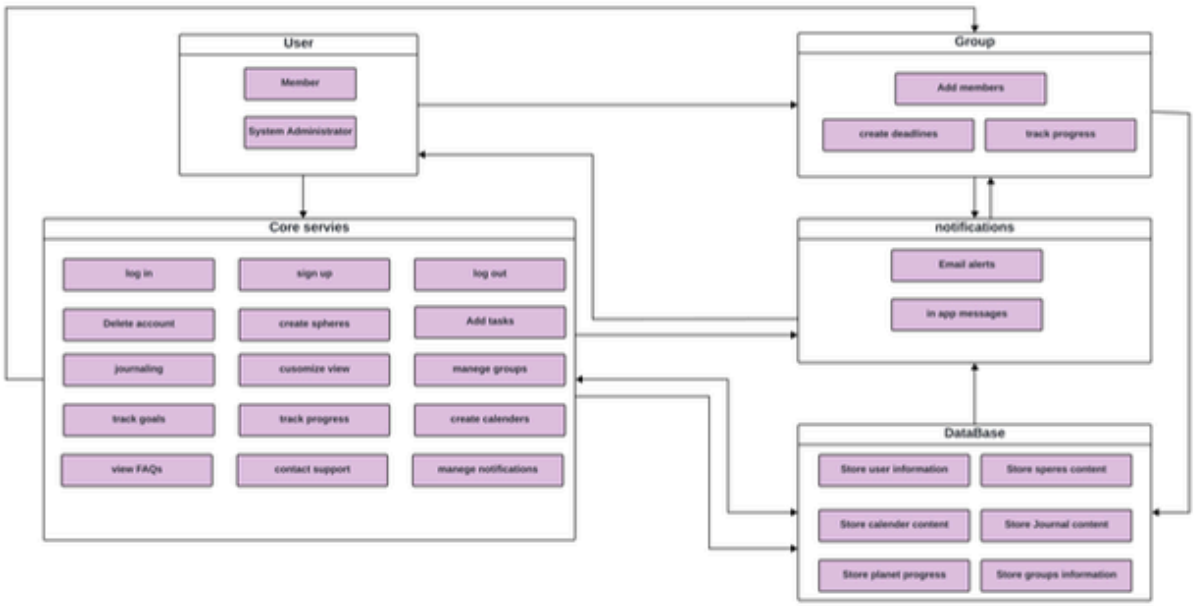
Subsystem model

BY ORGANIZING SPHERIS 24/7 INTO CLEARLY DEFINED SUBSYSTEMS USING **OBJECT-ORIENTED MODELING**, WE ACHIEVE A HIGHLY MODULAR AND MAINTAINABLE ARCHITECTURE, WHERE ACH SUBSYSTEM ENCAPSULATES A SPECIFIC RESPONSIBILITY- SUCH AS USER MANAGEMENT, CORE FUNCTIONALITIES, OR NOTIFICATIONS MAKING THE SYSTEM EASIER TO DEVELOP, TEST, AND SCALE. THIS SEPARATION OF CONCERNS NOT ONLY REDUCES INTERDEPENDENCIES BUT ALSO SUPPORTS COLLABORATIVE DEVELOPMENT, WHERE TEAMS CAN WORK ON DIFFERENT MODULES SIMULTANEOUSLY. MOREOVER, OBJECT-ORIENTED PRINCIPLES LIKE INHERITANCE AND ENCAPSULATION ALLOW US TO REUSE COMPONENTS ACROSS FEATURES (E.G., TASK TRACKING FOR INDIVIDUALS AND GROUPS), PROMOTING CONSISTENCY AND REDUCING REDUNDANCY. SO OVERALL, THIS STRUCTURE ENHANCES THE FLEXIBILITY, SCALABILITY, AND ROBUSTNESS OF SPHERIS 24/7, ENSURING IT GROWS WITH USERS' PRODUCTIVITY NEED

CONTROL MODEL

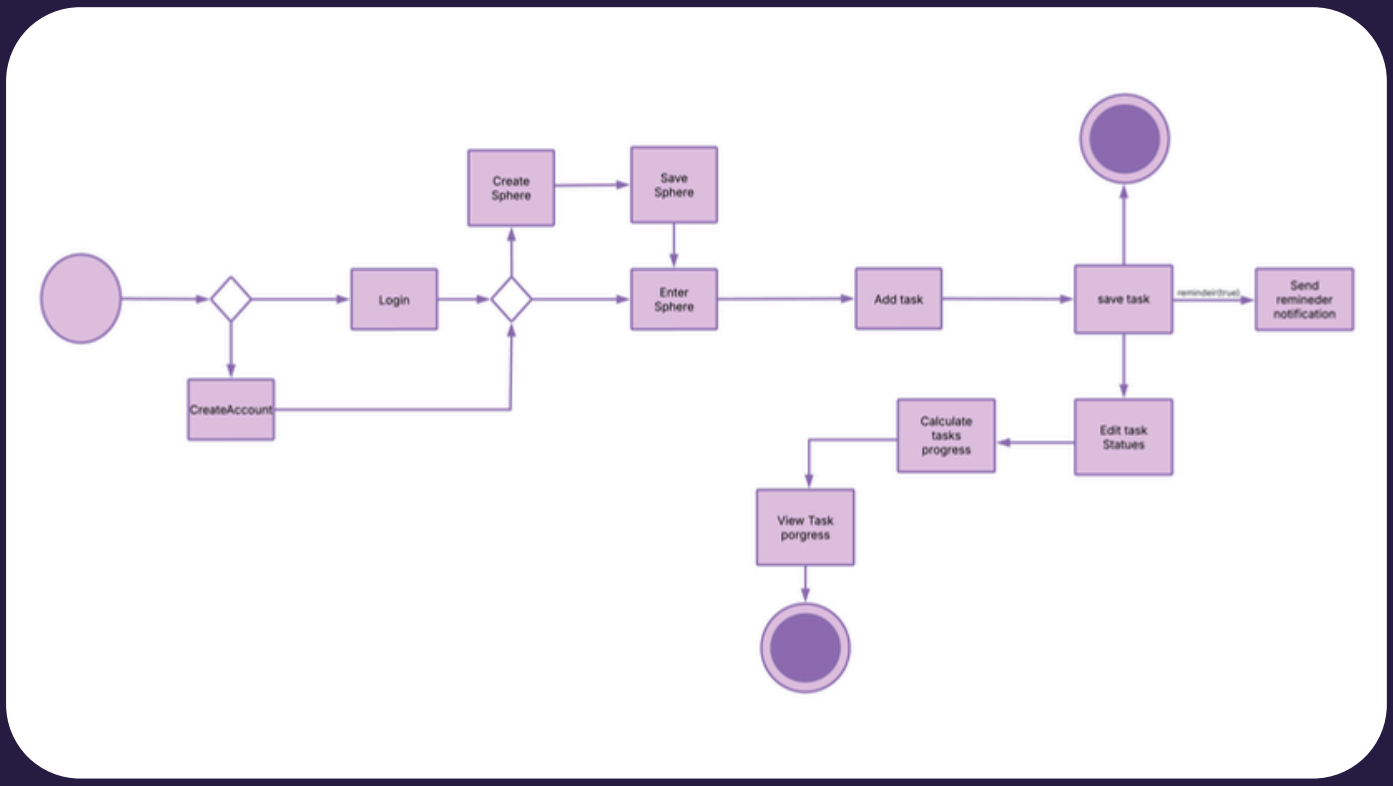
FOR THE CONTROL MODEL SPHERIS 24/7 ADOPTS AN **EVENT-DRIVEN BROADCASTING**, ENABLING SEAMLESS AND DECOUPLED COMMUNICATION BETWEEN SUBSYSTEMS. WHEN A USER OR GROUP PERFORMS AN ACTION SUCH AS COMPLETING A TASK, UPDATING PROGRESS, OR MANAGING DEADLINES AN EVENT IS BROADCAST ACROSS THE SYSTEM. SUBSYSTEMS LIKE NOTIFICATIONS LISTEN FOR THESE EVENTS AND RESPOND ACCORDINGLY BY GENERATING REAL-TIME UPDATES, ALERTS, OR IN-APP MESSAGES.
THIS MODEL ENHANCES SYSTEM RESPONSIVENESS, SUPPORTS SCALABILITY, AND ENSURES THAT FEATURES LIKE PROGRESS TRACKING, GROUP COLLABORATION, AND JOURNALING CAN OPERATE INDEPENDENTLY WHILE STAYING SYNCHRONIZED.

ARCHITECTURAL DIAGRAM
BLOCK DIAGRAM



SPHERIS 24/7

5.5.1
BEHAVIORAL DIAGRAM
STATE DIAGRAM



SPHERIS 24/7

6.1 DESIGN OF TESTS

UNIT TESTING

SPHERIS 24/7

TEST CASE ID	USE CASE ID	DESCRIPTION	TEST DATA/ INPUT	EXPECTED RESULT	PASS/FAIL CRITERIA	ACTUAL RESULT	ANY COMMENTS
TC1	UC1	Tests if the user can create Account	User entered valid information (name, email, password)	Account is successfully created	Pass if the account is created and stored. Fail if account creation fails	As Expected	None
TC2	UC2	Tests if the user can delete their account	User clicks on “Delete Account” after logging in	Account is permanently deleted from the system	Pass if the account is removed and user no longer has access. Fail if the account still exists or deletion fails.	As Expected	None
TC3	UC3	Tests if the user can edit their account details	User updates their name and profile picture, then clicks “Save”	Updated information is saved and reflected in the user’s account	Pass if the changes are saved and visible. Fail if changes are not applied	As Expected	None
TC4	UC4	Tests if the user can log in with correct credentials	User enters registered email and correct password	User is successfully logged in and redirected to the homepage	Pass if login is successful and access is granted. Fail if login is denied	As Expected	None
TC5	UC5	Tests if the user can log out successfully	User clicks the “Logout” button while logged in	User is logged out and redirected to the login screen	Pass if session ends and user is taken back to login. Fail if user remains logged in	As Expected	None
TC6	UC6	Tests if the user can customize their view preferences	User selects preferred mode/ page theme and rearranges layout sections, then saves changes	User interface updates according to selected preferences	Pass if preferences are saved and applied immediately. Fail if changes are not reflected or reset	As Expected	None
TC7	UC7	Tests if the user can create a new journal entry	User clicks “New Journal”, enters a title and content, then saves	A new journal entry is created and saved in the user’s journal list	Pass if the journal appears in the list with correct data. Fail if it's not saved or content is missing	As Expected	None

6.1
DESIGN OF TESTS
UNIT TESTING

SPHERIS 24/7

TEST CASE ID	USE CASE ID	DESCRIPTION	TEST DATA/ INPUT	EXPECTED RESULT	PASS/FAIL CRITERIA	ACTUAL RESULT	ANY COMMENTS
TC8	UC8	Tests if the user can add content to an existing journal entry	User opens a journal entry, writes new text content, then clicks “Save”	New content is saved and appears within the journal entry	Pass if content is correctly saved and visible. Fail if content is lost or not saved	As Expected	None
TC9	UC9	Tests if the user can edit an existing journal entry	User selects a journal entry, updates the title and content, then saves changes	The journal entry is updated with the new content	Pass if the updated content is saved and shown correctly. Fail if changes are not saved or content remains unchanged	As Expected	None
TC10	UC10	Tests if the user can add a new Spheris to their environment	User clicks “Add Spheris”, enters Spheris name and selects type, then confirms	A new Spheris is created and appears in the user’s environment	Pass if the Spheris is added and visible with correct details. Fail if it's not added or data is missing	As Expected	None
TC11	UC11	Tests if the user can edit an existing Spheris	User selects a Spheris, changes its name and updates its type, then clicks “Save”	The selected Spheris is updated with the new information	Pass if changes are saved and reflected in the environment. Fail if updates are not applied	As Expected	None
TC12	UC12	Tests if the user can delete a Spheris from their account	User selects a Spheris and clicks “Delete”	The selected Spheris is removed from the environment and no longer visible	Pass if the Spheris is successfully deleted. Fail if the Spheris remains or is not removed.	As Expected	None
TC13	UC13	Tests if the user can add a new task within a sphere and trigger an email for special restrictions	User selects a sphere, adds a task with a due date, priority, and special restriction, then clicks “Save”	Task is added to the sphere, and an email is sent to the user regarding the restriction	Pass if the task is added and email is successfully sent. Fail if task is not added or email is not triggered	As Expected	None
TC14	UC14	Tests if the user can modify an existing task (such as changing details or deadlines)	User selects a task, updates the task details or deadline, then clicks “Save”	Task is updated with the new details and deadline	Pass if the task is updated and reflects the new information. Fail if updates are not saved or not visible.	As Expected	None

6.1
DESIGN OF TESTS
UNIT TESTING

TEST CASE ID	USE CASE ID	DESCRIPTION	TEST DATA/ INPUT	EXPECTED RESULT	PASS/FAIL CRITERIA	ACTUAL RESULT	ANY COMMENTS
TC22	UC22	Tests if the user can view the calendar on a yearly basis.	User selects the yearly view option on the calendar.	The calendar is displayed in a yearly format, showing all months and relevant tasks, events, and deadlines for the year.	*Pass if the calendar is displayed correctly in the yearly format. Fail if the view does not switch to a yearly format or if data is missing.*	As Expected	None
TC23	UC23	Tests if the user can view the calendar on a monthly basis.	User selects the monthly view option on the calendar.	The calendar is displayed in a monthly format, showing all days and relevant tasks, events, and deadlines for the selected month.	*Pass if the calendar is displayed correctly in the monthly format. Fail if the view does not switch to a monthly format or if data is missing.*	As Expected	None
TC24	UC24	Tests if the user can view the calendar on a daily basis.	User selects the daily view option on the calendar.	The calendar is displayed in a daily format, showing all events, tasks, and deadlines for the selected day.	*Pass if the calendar is displayed correctly in the daily format. Fail if the view does not switch to a daily format or if data is missing.*	As Expected	None
TC25	UC25	Tests if the user can search for another user.	User enters a username in the search bar and submits the search.	The system displays the profile or information of the user that matches the entered username.	*Pass if the correct user is found and their details are displayed. Fail if the system doesn't find the user or displays incorrect information.*	As expected	None
TC26	UC26	Tests if the user can travel from one planet to another in the planetarium based on working hours.	User selects a planet to travel to and specifies the working hours.	The user travels to the selected planet, and the system updates the user's progress based on the working hours.	*Pass if the user successfully travels to the next planet and their progress is updated correctly. Fail if the travel doesn't occur or the progress isn't updated.*	As expected	None
TC27	UC27	Tests if the user can view the entire planetarium, including its components (planet, planet description, galaxy squad, and stellar journal).	User selects the option to view the planetarium.	The planetarium is displayed, showing the planets, their descriptions, the galaxy squad information, and the stellar journal.	*Pass if the planetarium is displayed correctly with all components (planet, description, galaxy squad, stellar journal) visible. Fail if any components are missing or not displayed properly.*	As expected	None
TC28	US28	Tests if the system provides help and FAQs to assist users with common issues.	User selects the "Help & FAQs" option from the system's interface.	The system displays a list of frequently asked questions and help topics to assist the user with common issues or inquiries.	*Pass if the help and FAQ information is displayed correctly and provides relevant answers or guidance. Fail if the help section is empty or contains incorrect information.*	As expected	None

6.2
DESIGN OF TESTS
INTEGRATION TESTING

TEST CASE ID	TEST SCENARIO	INTEGRATED MODULES	DESCRIPTION	STEPS	PASS/FAIL CRITERIA	EXPECTED RESULT	ACTUAL RESULT
TC29	Deleting a User Account and Associated Data	User Management, Task Management, Goal Management	This test checks if deleting a user account properly removes all their related data, including tasks and goals.	• Log in with a test account • Add a few tasks and goals • Delete the account	*Pass if: No data is left behind after deletion* *Fail if: Any tasks or goals are still in the system*	The account and all its tasks/goals should be completely removed	As Expected
TC30	Task Updates Reflecting in Calendar	Task Management, Calendar	Verifies that when a task is updated (status or due date), the calendar reflects the changes.	• Update an existing task's status and due date • Open the calendar	*Pass if: Changes show up properly in the calendar* *Fail if: Calendar shows outdated or wrong info*	The updated task should appear with the correct date and status	As Expected
TC31	Completed Goals Sync with Planetarium	Goal Management, Planetarium	Makes sure that when a user finishes a goal, their progress updates visually in the planetarium.	• Complete a goal • Go to the planetarium	*Pass if: New achievement is shown* *Fail if: Nothing changes*	Planetarium visuals should update to show progress	As Expected
TC32	Calendar Consistency Between Views	Task Management, Calendar (Daily, Monthly, Yearly)	This test checks if all calendar views display the same data without mismatch.	Steps: • Add tasks with deadlines • Check all calendar views	*Pass if: No missing or wrong entries* *Fail if: Inconsistencies between views*	Same tasks should show up correctly in all views	As Expected
TC33	Editing a Goal Updates the Goal List	Goal Editor, Goal List	Checks if goal edits (like changing title or description) are saved and shown in the list right away	Steps: • Edit a goal • View the goal list	*Pass if: Edits are reflected* *Fail if: Old data still appears*	Updated version should be listed	As Expected
TC34	User Search and Profile Display	Search Bar, User Profile Page	Makes sure that searching for a username brings up the correct profile info	Steps: • Enter a valid username • Check the result	*Pass if: Correct profile is found* *Fail if: No result or wrong user appears*	User profile should show up with correct details	As Expected

6.3
ACCEPTANCE TESTING

After completing all core functionalities of Spheris and ensuring their stability through functional and integration testing, we allowed users to explore the system and share their feedback. Most users were able to navigate the system smoothly and complete tasks like adding goals, tracking progress, exploring the calendar, and viewing the planetarium without major issues.

One user mentioned that they initially felt overwhelmed when interacting with the planetarium due to the many components it included. To improve the experience, we plan to provide a guided walkthrough on first launch, especially highlighting features like traveling between planets and interpreting progress updates.

Another point of feedback involved the visibility of tasks and goals when switching between calendar views. Users expressed that while the system functions correctly, a tooltip or legend could make it clearer what each color or symbol represents.

We take this kind of feedback seriously and continuously work on making Spheris more user-friendly. We're committed to improving the user experience based on real interactions, ensuring the system not only meets the requirements but also adapts to user needs as they evolve.

7 REFERENCES

[1] TickTick, “TickTick: Task & To-Do List,” [Online]. Available: https://www.ticktick.com. [Accessed: Jan. 30, 2025].

[2] Todoist, “Todoist: To-Do List and Task Manager,” [Online]. Available: https://www.todoist.com. [Accessed: Jan. 30, 2025].

[3] Notion, “Notion: All-in-One Workspace,” [Online]. Available: https://www.notion.so. [Accessed: Jan. 30, 2025].