YEDİTEPE UNIVERSITY

T.C.

YEDITEPE UNIVERSITY

DEPARTMENT OF INDUSTRIAL ENGINEERING

ISE 402 – CSE 344 INTERDICIPLINARY SYSTEM ANALYSIS PROJECT

Prepared by:

ISE

Berrak Şevval KONCA
Lale Arzü Elena KARABULUT
Tuğçe TEPECİKLİOĞLU
Emir Zahid PARTO

CSE

Arda IRMAK
Asude Beyza DEMİRBOĞA
Fevzi BABAOĞLU
Hatice Müberra GÜL

INDEX

1. INTRODUCTION

1.1. PURPOSE

Financial instability impacts millions of individuals. This pervasive issue can lead to personal distress, debt accumulation, economic setbacks, and societal strain.

Prior to facing financial crises, individuals should have access to resources that help them understand their financial standing and identify safe financial practices. These preparations are essential for mitigating challenges that may arise during times of financial difficulty. Additionally, the critical phase of managing financial crises lies in the aftermath, where coordination and communication play a pivotal role in facilitating recovery efforts.

Our financial management app is dedicated to enhancing users' financial well-being by providing accessible tools for saving money and planning future expenses. With our user-friendly interface, individuals can effortlessly track their spending habits and visualize their financial plans, allowing for better control over their finances. Additionally, our app offers a streamlined approach to managing subscriptions and future payments, making it easy for users to monitor and adjust their recurring expenses. Through these features, our aim is to empower users to take charge of their finances, save money effectively, and achieve their financial goals with confidence.

In summary, our financial management app is designed to bolster financial stability and resilience, particularly in times of economic uncertainty. By providing users with effective tools for financial planning, intervention, and recovery, our app aims to minimize financial risks and facilitate smoother financial recoveries for individuals.

In this document, we aimed to explain the problem by using requirements engineering and the solution by using architectural diagrams to fulfill the needs of its users.

1.2. BACKGROUND

Financial management involves budgeting, saving, investing, and managing debts to ensure financial security and well-being. It plays a vital role in individuals' lives, influencing their ability to afford basic needs, pursue education, plan for retirement, and respond to unexpected financial challenges.

In response to financial challenges, individuals and organizations must prioritize financial preparedness and resilience. Developing financial literacy, establishing emergency funds, and creating contingency plans are essential steps in navigating financial storms and building a secure financial future.

Our financial management app aims to empower individuals to take control of their finances, navigate financial challenges, and achieve their financial aspirations. Through features such as expense tracking, subscription/future payment management, calendar to track their incomes/expenses and events, our app provides users with the tools and resources they need to manage their money effectively.

1.3. MOTIVATION

Unlike current money management apps, this application helps users to plan and monitor not only their current money flow but also their future money flow. This app enables users to manage their finances more efficiently and effectively. Users can forecast their future income and expenses, allowing them to plan their budgets more accurately and prepare for potential financial challenges. This way, they can take more informed steps towards achieving their financial goals. The application provides detailed analyses of users' financial situations and helps them develop long-term financial strategies, thereby enhancing their financial security and reducing stress levels.

Statement of problems with the existing system

Lack of Comprehensive Visibility: The management of personal finances often relies on disparate tools and manual methods, lacking a centralized platform accessible to all users. This fragmentation makes it challenging for individuals to gain a comprehensive view of their financial health and track their transactions effectively.

Inefficient Resource Allocation: Similar to the improper distribution of emergency materials in the existing emergency management system, individuals may struggle with allocating their financial resources optimally. Without clear insights into their spending patterns and financial goals, they may overspend in certain areas while neglecting others, leading to financial imbalance and uncertainty.

Lack of Support for Diverse Needs: Individuals with varying financial circumstances, such as freelancers, small business owners, or retirees, may require tailored financial solutions and support that are not readily available in existing systems.

Overall, the existing financial management systems face challenges related to visibility, resource allocation, preparedness measures, and support for diverse needs. Addressing these issues requires the development of comprehensive, accessible, and user-centric financial management solutions that empower individuals to navigate their financial journeys effectively.

The New System

Centralized Platform: Our system offers a centralized platform accessible to all users, providing a holistic view of their financial health. Through a user-friendly interface, individuals can track their income, expenses, savings, investments, and debts in one place, eliminating the need for disparate tools and manual methods.

Budgeting and Expense Tracking: To tackle inefficient resource allocation, our system includes robust budgeting and expense tracking tools. Users can set personalized budgets, categorize their expenses, and monitor their spending patterns in real-time. This helps individuals identify areas of overspending and adjust their financial habits accordingly, promoting financial balance and stability.

Accessibility and Inclusivity: Much like the need for support for diverse needs in emergency management, our system is designed to be accessible and inclusive to users from all backgrounds. We offer customizable features to accommodate different financial circumstances, such as variable income streams, irregular expenses, and unique financial goals. Additionally, our system prioritizes user privacy and security, ensuring a safe and inclusive environment for all users.

Overall, our new financial management system addresses the limitations of existing systems by providing users with centralized access, robust budgeting tools, and inclusivity. By empowering individuals to take control of their finances and make informed decisions, our system aims to promote financial stability and resilience for all users.

1.4. PURPOSE OF THE SYSTEM

We have designed our system with a primary focus on effectively empowering users to manage their finances by offering a user-friendly platform.

Unlike common apps, we provide a comprehensive solution for budget management, placing a strong emphasis on managing future expenses (such as bills, installments, and planned expenses) alongside tracking incomes and expenses. We aim to simplify financial tracking across multiple accounts, offering clear insights into spending patterns and helping users make informed decisions about their money. Moreover, our calendar feature provides a holistic view of users' financials, displaying income, expenses, events, and upcoming expenses in one unified platform.

Ultimately, our goal is to promote financial awareness and stability with ease of use for individuals at all levels of financial literacy, regardless of banking affiliations.

2. REQUIREMENTS

2.1. FUNCTIONAL REQUIREMENTS

2.1.1. Description of the system functionalities

- Users can input their income sources.
- Users can input their salaries.
- Users can input their freelance earnings.
- Users can input their other sources of revenue.
- Expenses can be recorded.
- Expenses can be categorized by expense categories.
- Users have the option to manually input transactions.
- Users receive notifications/alerts.
- The app employs robust security measures to protect user data.

By offering these comprehensive functionalities, the financial management app provides users with powerful tools to effectively manage their finances, achieve their financial goals, and make informed financial decisions.

1.1.1. Description of the system users

Individual Users

- This category includes individuals from various demographics who want to manage their personal finances effectively.
- Users may have different financial goals.
- Users may have different priorities.
- Individual users may be beginners seeking basic financial guidance.
- Individual users may be experienced investors looking for advanced portfolio management tools.
-

Small Business Owners

- Small business owners form another user group who need tools to track their business finances.
- Small business owners form another user group who need tools to manage their business finances.
- These users may include proprietors.
- These users may include freelancers.
- These users may include entrepreneurs running small businesses.
- Small business owners require features tracking income and expenses.
- Small business owners require features invoicing clients.
- Small business owners require features managing business accounts.
- Small business owners require features generating financial reports for tax purposes.

- Small business owners require features generating financial reports for business planning.

Students

- Students represent a significant user segment who are managing limited budgets while studying.
- These users may need tools to track expenses related to tuition.
- These users may need tools to track expenses related to textbooks.
- These users may need tools to track expenses related to home expenses.
- These users may need tools to track expenses related to food.
- These users may need tools to track expenses related to transportation.
- These users may need tools to track expenses related to other student-related expenses.
- Students may benefit from features tailored to managing student loans.
- Students may benefit from features tailored to tracking part-time job income.

Freelancers

- Freelancers require specialized tools to manage their irregular income streams.
- Freelancers require specialized tools to manage their irregular expenses.
- Independent contractors require specialized tools to manage their irregular income streams.
- Independent contractors require specialized tools to manage their irregular expenses.
- Freelancers prioritize features that help them optimize their cash flow.
- Freelancers prioritize features that help them optimize their track billable hours.

Families

- Families consist of multiple individuals who share financial responsibilities.
- Families consist of multiple individuals who share financial goals.
- Family users may benefit from the feature of budget pooling.
- Family users may benefit from the feature of expense splitting.
- Family users may benefit from the feature of shared financial calendars to coordinate household finances.

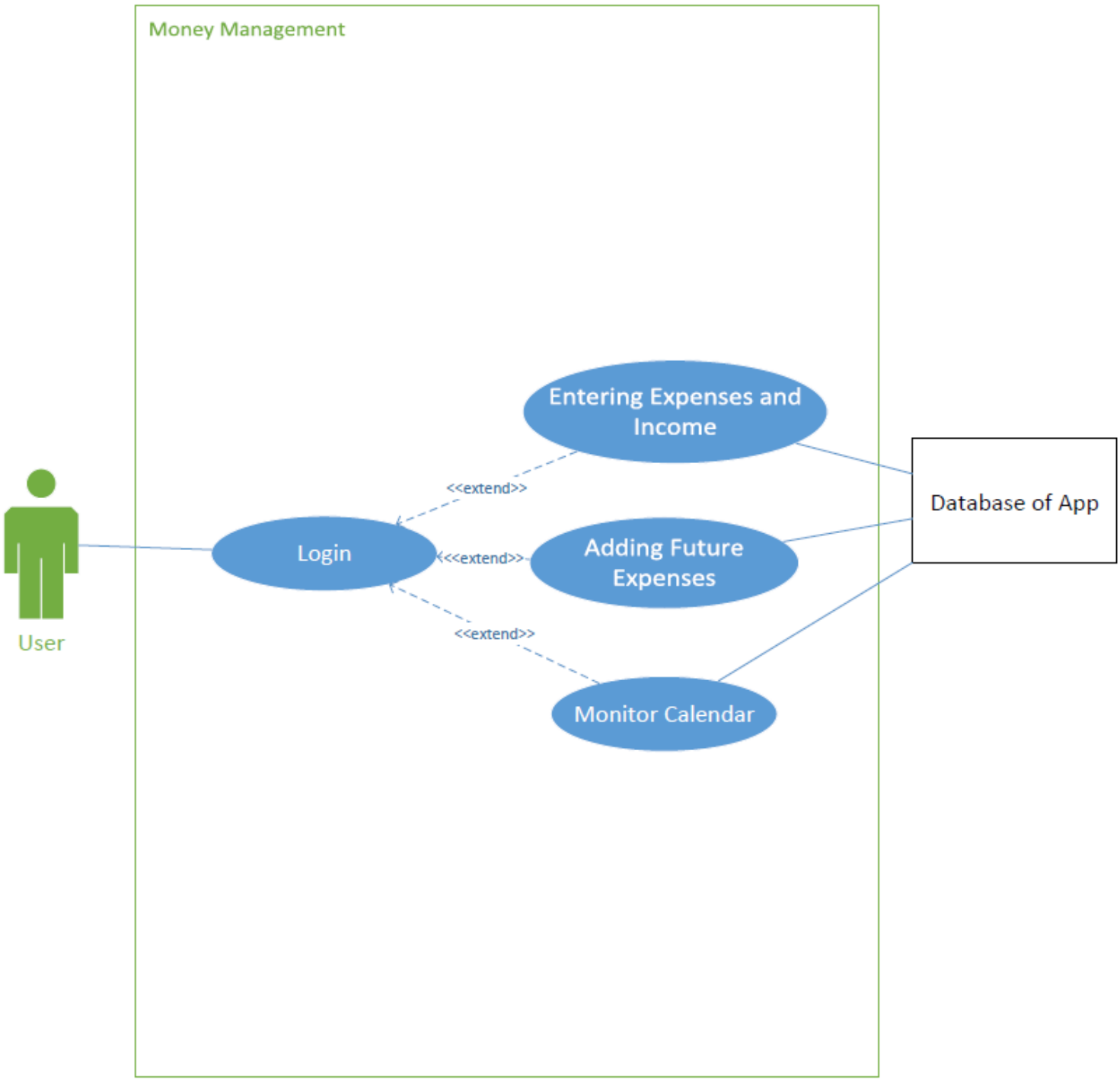
2.2. SPECIFIC REQUIREMENTS:

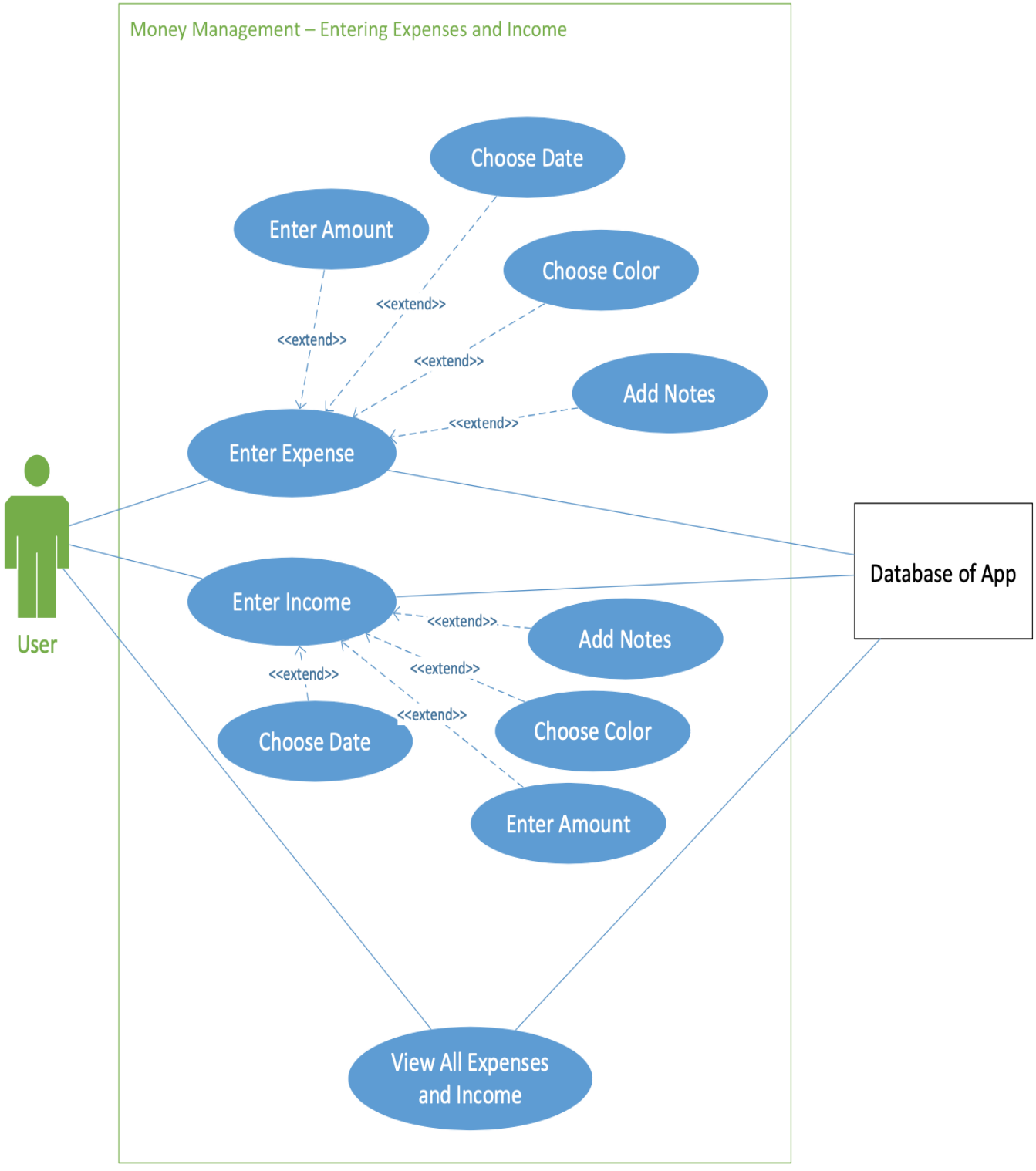
- The system shall provide a registration process for new users to create accounts.
- The system shall display login and sign up buttons to enter the interface.
- The system shall request information on the sign up page username, e-mail, password to create accounts.
- The system shall enable users to reset their passwords if users forgot their passwords.
- The system shall link passwords and usernames with personalized e-mails.
- The system shall provide a validation process to ensure that entered information matches with the specified criteria.

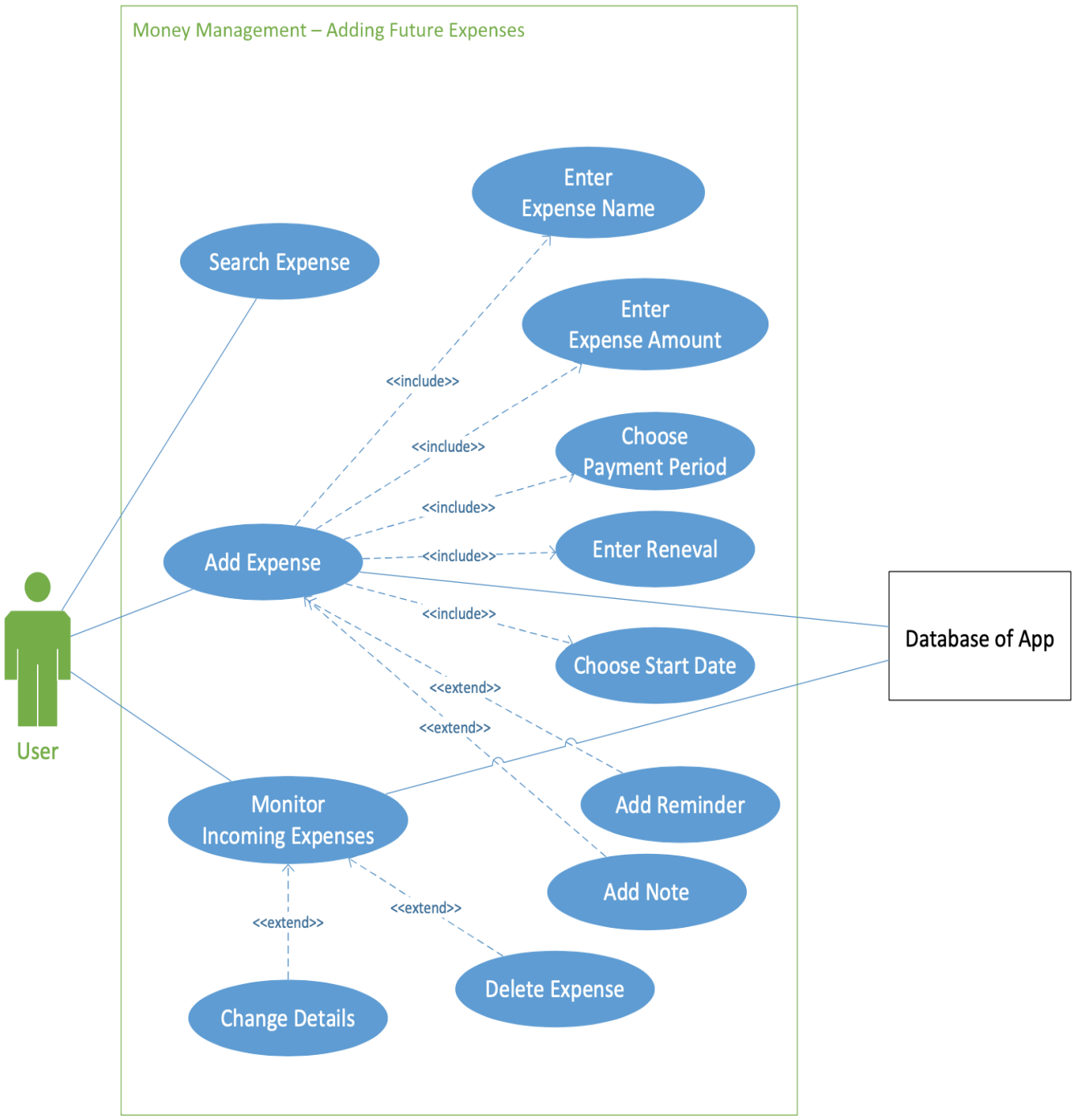
- The system shall enable users to be able to log in using their registered email address or username along with their password.
- The system shall store users' personal data, including name, email address, password.
- The system should operate quickly and be protected against security vulnerabilities.
- The system shall enforce access controls to prevent unauthorized users from modifying the contents.
- The system shall manage the password reset process by including verification steps to confirm the identity of the user.
- The system shall have the option to manually log out of their accounts to end their session.
- The system shall provide appropriate error messages for failed login attempts.
- The system shall provide clear instructions to users on how to resolve login issues for resetting their password.
- The system shall provide clear instructions to users on how to resolve login issues for contacting support.
- The system shall provide a menu in order to show the pages included to the system interface.
- The system shall enable users to enter a future payment.
- The system shall enable users to enter an income.
- The system shall enable users to enter an expense.
- The system shall enable users to enter an event.
- The system shall open the calendar after entering the interface.
- The calendar system must include a feature to display a monthly view.
- The calendar system shall enable users to switch the year considered.
- The calendar system shall highlight the current day.
- The calendar system shall enable users to monitor events.
- The calendar system shall enable users to monitor expenses.
- The calendar system shall enable users to monitor incomes.
- The calendar system shall enable users to monitor future payments.
- The calendar system shall enable users to enter an event on the calendar.
- The calendar system shall enable users to monitor upcoming events for future dates.
- The system shall enable users to add events to the calendar.
- The system shall enable users to add expenses to the calendar.
- The system shall enable users to add incomes to the calendar.
- The system shall enable users to give a specific name according to their events.
- The system shall enable users to choose a date while entering an event even if the considered event is not related to the current date.
- The system shall enable users to customize colors based on type.
- The system shall enable users to save the changes they made.
- The system shall provide users to add future expenses.
- The system shall display the user profile on every page.
- The system shall display the hamburger bar section on every page.
- The system shall display upcoming expenses to give an insight to the user.
- The system shall enable users to search for their upcoming payments.
- The system shall enable users to list expenses.
- The system shall enable users to list incomes.

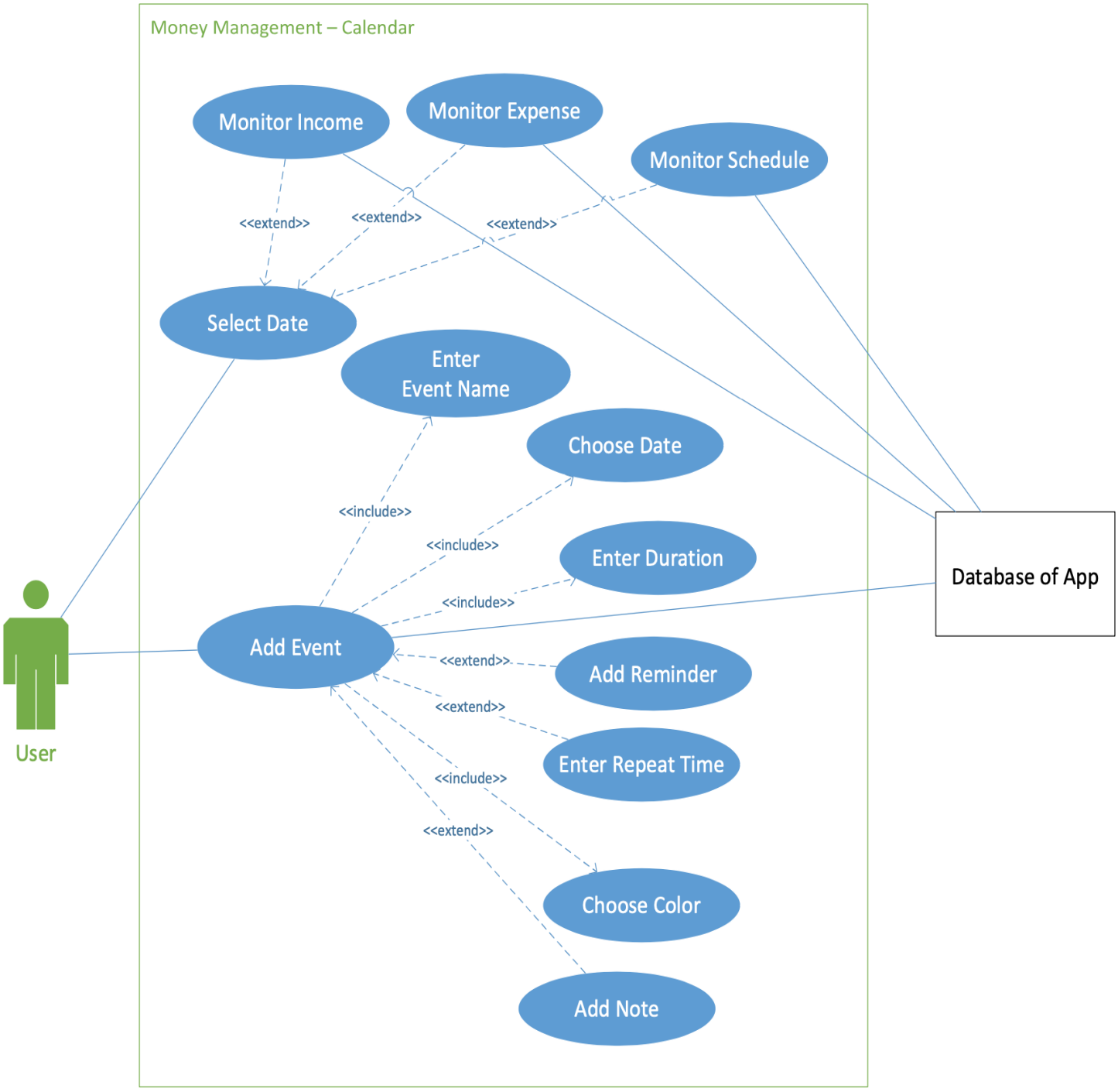
- The system shall enable users to list events.
- The upcoming events should be shown in a descending order based on their due dates.
- The system shall enable users to delete the future expense.
- The system shall enable users to get information about the future expense.
- The system shall enable users to choose which balance they are using while entering an income.
- The system shall enable users to choose which balance they are using while entering an expense.
- The system shall enable users to categorize their expenses.
- The system shall enable users to categorize their incomes.
- The system shall allow users to enter an amount.
- The system shall allow users to enter notes about their expenses.
- The system shall allow users to enter notes about their incomes.
- The system shall allow users to enter notes about their events.
- The system shall exhibit every expense and income on the calendar within those specific dates.
- The system shall allow users to select time durations while entering a new event on the calendar.
- The system shall enable users to add upcoming expenses.
- The system shall enable users to add upcoming events.
- The system shall enable users to delete the changes they have made.
- The Hamburger Bar shall enable users to have shortcuts for Settings, Support and Log out sections.
- The system shall display the current budget.
- The system shall enable users to monitor their incomes.
- The system shall enable users to monitor their expenses.
- The system shall enable users to monitor their total situation.
- The system shall enable users to enter the amount paid.
- The system shall enable users to enter the amount gained.
- The system shall enable users to add captions about the entered income.
- The system shall enable users to add captions about the entered expense.

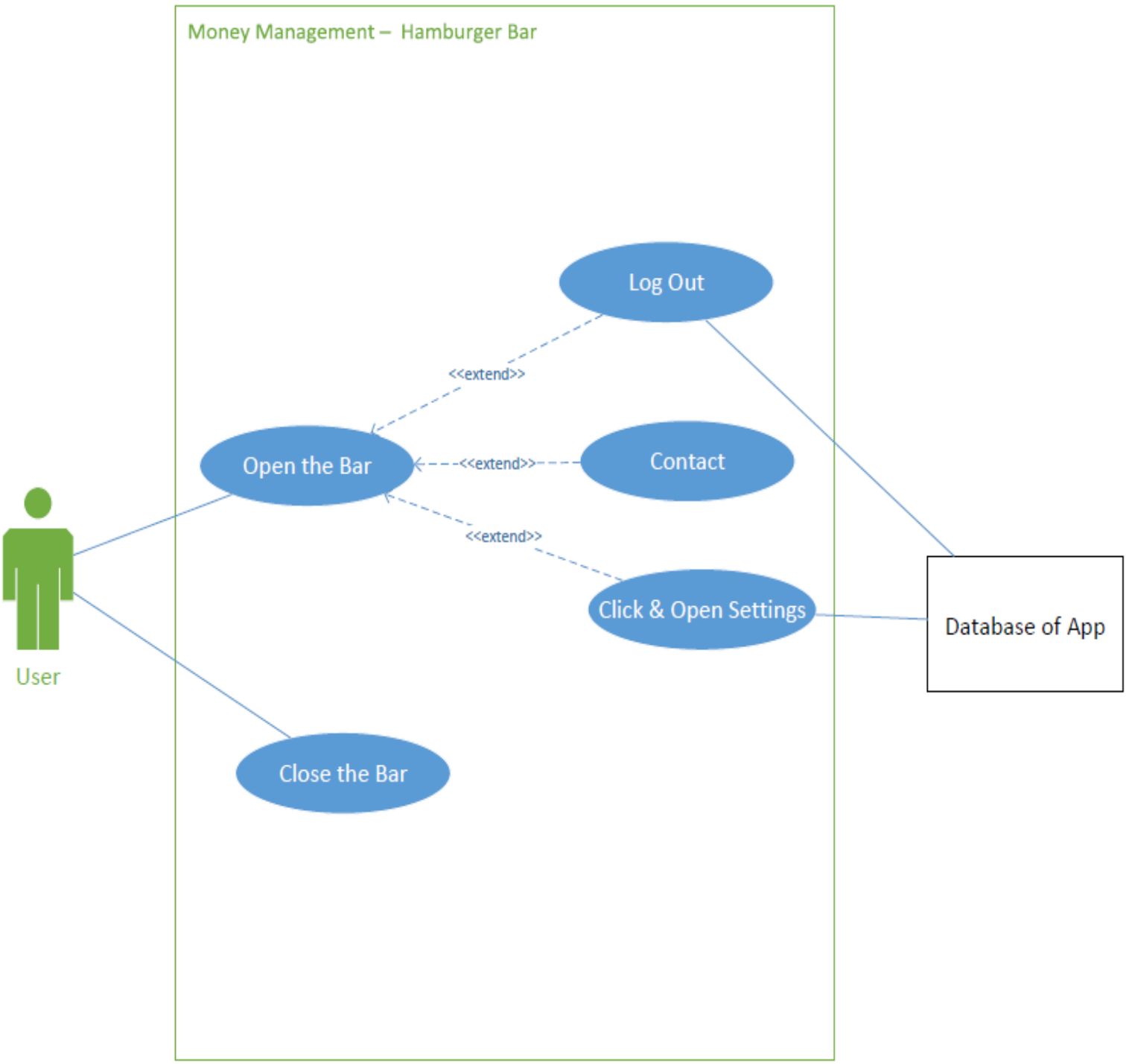
2.3. USE CASE DIAGRAMS

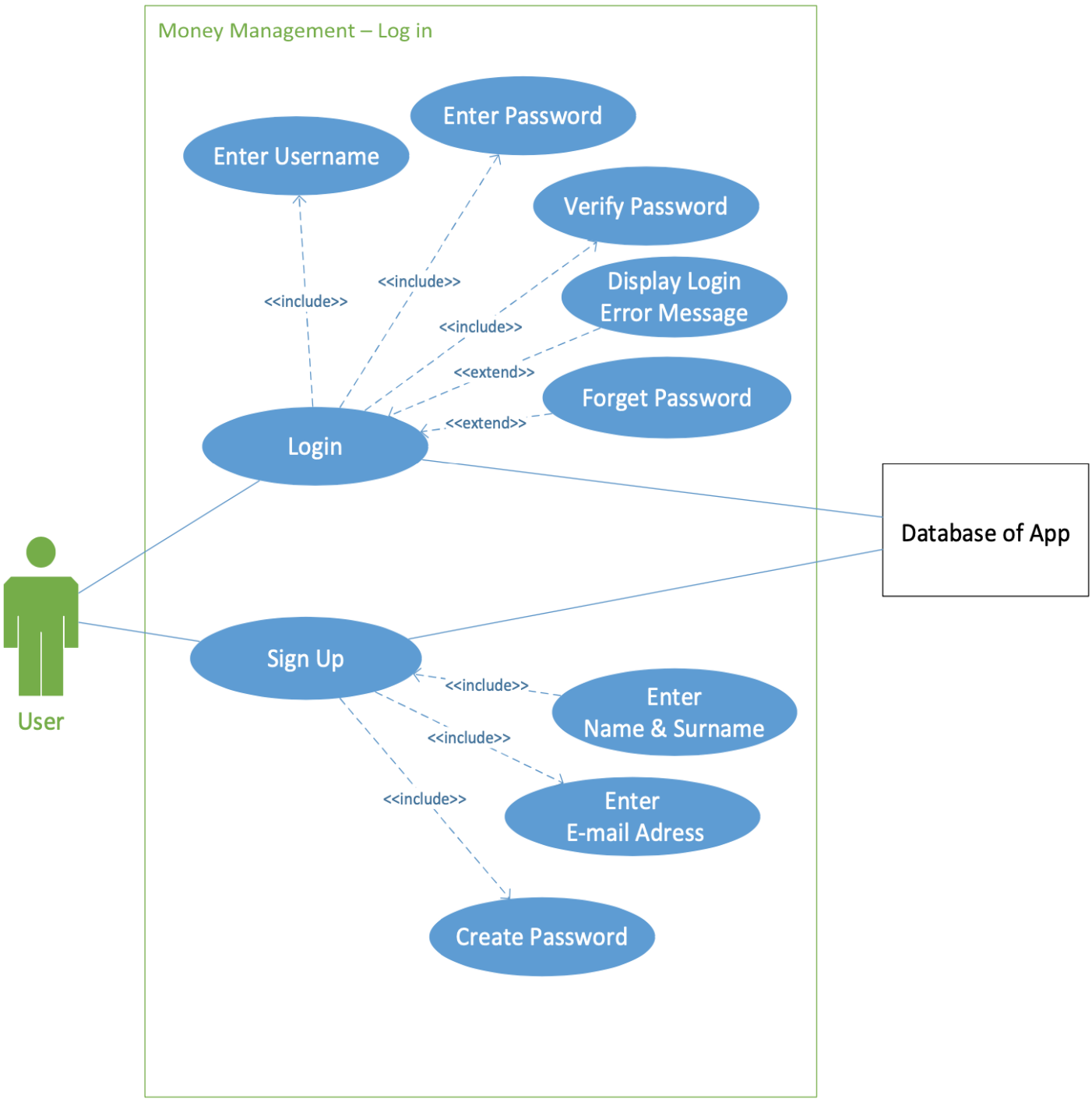



Money Management – Entering Expenses and Income
Choose Date
Enter Amount
Choose Color
<<extend>>
<<extend>>
<<extend>>
Add Notes
<<extend>>
Enter Expense
User
Database of App
Enter Income
<<extend>>
Add Notes
<<extend>>
<<extend>>
Choose Date
<<extend>>
Choose Color
Enter Amount
View All Expenses and Income


Money Management – Adding Future Expenses
Enter
Expense Name
Search Expense
Enter
Expense Amount
<<include>>
<<include>>
Choose
Payment Period
<<include>>
Add Expense
<<include>>
Enter Reneval
<<include>>
Choose Start Date
Database of App
<<extend>>
<<extend>>
User
Monitor
Incoming Expenses
Add Reminder
Add Note
<<extend>>
<<extend>>
Delete Expense
Change Details


Money Management – Calendar
Monitor Income
Monitor Expense
Monitor Schedule
<<extend>>
<<extend>>
<<extend>>
Select Date
Enter
Event Name
Choose Date
<<include>>
<<include>>
Enter Duration
<<include>>
Add Event
<<extend>>
Add Reminder
<<extend>>
Enter Repeat Time
<<include>>
<<extend>>
Choose Color
Add Note
Database of App
User


Money Management – Hamburger Bar
Log Out
<<extend>>
Open the Bar
<<extend>>
Contact
<<extend>>
Click & Open Settings
Database of App
User
Close the Bar


Money Management – Log in
Enter Password
Enter Username
Verify Password
Display Login
Error Message
Forget Password
Login
Sign Up
Enter
Name & Surname
Enter
E-mail Adress
Create Password
<<include>>
<<include>>
<<include>>
<<extend>>
<<extend>>
<<include>>
<<include>>
<<include>>
Database of App
User

2.4. USE CASE PRIORITY LIST

Use Case	Priority Rank	Reason
Log in	High	The system will not be able to function if no one is able to log in to the system.
Entering Expenses and Incomes	High	Entering expenses and incomes is a critical use case for a system to function.
Adding Future Expenses	High	Adding future expenses is a critical use case for a system to function.
Monitor Calendar	Medium	Monitoring calendar is important but does not affect any critical functionality of the system.
Open Hamburger Bar	Low	Opening hamburger bar does not affect any critical functionality.
Choose Date	High	Choosing date is a critical use case for the calendar system to show current expenses, incomes and events on a specific date.
View All Expenses and Incomes	High	Monitoring all expenses and incomes is a crucial use case to have insight about a specific date.
Add Notes	Low	Adding notes does not affect any critical functionality.
Choose Color	Medium	Choosing color is important but does not affect any critical functionality of the system.
Search Expense	High	Searching expense is a critical use case for the system to function.
Enter Expense Name	High	Entering expense name is important for collecting and clustering insight data.
Enter Expense Amount	High	Entering expense amount i important for monitoring the current budget.
Choose Payment Period	Medium	Choosing payment period is important but does not affect any critical functionality of the system.
Reminder	High	Adding reminder is crucial for user to stay disciplined.

Change Details	Medium	Changing details is important but does not affect any critical functionality of the system.
Delete Expense	Medium	Deleting expenses is important but does not affect any critical functionality of the system.
Enter Renewal	Medium	Entering renewal is important but does not affect any critical functionality of the system.
Add Event	Low	Adding events does not affect any critical functionality.
Enter Event Name	Low	Entering event name does not affect any critical functionality.
Enter Event Duration	Low	Entering event duration does not affect any critical functionality.
Click & Open the Bar	Low	Clicking & opening the bar is important but does not affect any critical functionality of the system.
Click & Close the Bar	Low	Clicking & closing the bar is important but does not affect any critical functionality of the system.
Log Out	High	Logging out is crucial for ensuring the security of user accounts and sensitive financial and personal information.
Contact	Low	Contact section does not affect any critical functionality.
Click & Open Settings	Medium	Clicking & opening the settings is important but does not affect any critical functionality of the system.
Change Username	Low	Changing username does not affect any critical functionality.
Enter Username	High	Entering username is fundamental for user authentication.
Enter Password	High	Entering password is an important for securely log in and authenticate user identities.

Verify Password	High	Verifying password enforces access controls and safeguard user data.
Display Log in Error Massage	Medium	Displaying log in error massage is important but does not affect any critical functionality of the system.
Forget Password?	High	Forgetting password allows users to regain access to their accounts quickly and efficiently.
Sign Up	High	Crucial for attracting new users and facilitate users entry into the system.
Enter Name & Surname	High	Crucial for personalize the given data by user.
Enter E-Mail Adress	High	Crucial for both personalize the given data by user date and accurately identify the user.
Create Password	High	Crucial for protecting users accounts from unauthorized access.

2.5. USE CASE SPECIFICATIONS

Use case ID: UC01	**Use Case:** Login
Primary Actors: User	
Preconditions: System is operational. User must open the login page and user should not be logged into the system.	
Main Flow: 1. User must enter the requested information (username and password). 2. User confirms the information that was entered by pressing Login Button. 3. Extension point: Display Login Error Message. 4. User gets notified that the login has been completed successfully.	
Post conditions: 1. User has been successfully logged into the system.	
Alternative Flow: Display Login Error Message.	
Preconditions: Information has been entered by the user does not match the system's rules or the information is false.	
Alternative Flow: The flow starts after main flow 3. 1. User gets notified by an error message which is returned.	
Post conditions: 1. User is forwarded back to the login screen.	

Use case ID: UC1.1.1	**Use Case:** Enter Username

Primary Actors: User

Preconditions: User has initiated the login process and is prompted to enter their username. The login screen is visible on the user interface.

Main Flow:

1. User navigates to the login screen.
2. System displays the username input field.
3. User enters their username into the designated input field.
4. User submits the entered username.
5. System verifies the username for correctness and completeness.
6. If the username is valid, the system proceeds to the next step (e.g., prompting for password entry). If the username is invalid or incomplete, an error message is displayed.
7. User interface transitions to the next step in the login process or displays an error message based on the validation result.

Post conditions:

If the entered username is valid and complete, the system proceeds to the next step in the login process.

If the entered username is invalid or incomplete, an error message is displayed, and the user remains on the login screen.

User interface state changes based on the validation result and the progression of the login process.

Alternative Flow: Username Not Found

Preconditions: User has initiated the login process and entered their username.
The entered username does not match any existing user records in the system.

Alternative Flow:

1. System verifies the entered username.
2. System does not find a match for the entered username in the user database.
3. An error message is displayed indicating that the username was not found.
4. User is prompted to re-enter their username or register if they don't have an account.

Post conditions: An error message is displayed indicating that the username was not found.
User is prompted to re-enter their username or register if they don't have an account.
User interface remains on the login screen awaiting further input.

Use case ID: UC1.1.2	**Use Case:** Enter Password
Primary Actors: User	
Preconditions: User has initiated the login process. User has entered a valid username.	
Main Flow: 1. System prompts the user to enter their password. 2. User enters their password. 3. If the password is correct, the system grants access to the user. 4. If the password is incorrect, the system displays an error message and prompts the user to re-enter their password.	
Post conditions: If the password is correct, the user gains access to the system and is redirected to the dashboard or main interface.	
Alternative Flow: Forgot Password	
Preconditions: User has initiated the login process. User has forgotten their password.	
Alternative Flow: 1. User clicks on the "Forgot Password" link. 2. System prompts the user to enter their email address. 3. User enters their email address. 4. System verifies the email address against the registered accounts. 5. If the email address is found, the system sends a password reset link to the user's email. 6. User checks their email and clicks on the password reset link. 7. System prompts the user to enter a new password. 8. User enters a new password and confirms it. 9. System updates the password in the database. 10. User is redirected to the login screen to log in with the new password.	
Post conditions: User successfully resets their password and can now log in with the new password. If the email address is not found or the password reset link is expired, the user is prompted to contact support for assistance.	

Use case ID: UC1.1.3	**Use Case:** Verify Password

Primary Actors: User

Preconditions: User has initiated the login process.
User has entered their username/email and password.

Main Flow:

1. User enters their username/email and password.
2. System verifies the entered username/email against the registered accounts.
3. If the username/email is found, the system retrieves the corresponding password from the database.
4. System compares the entered password with the retrieved password.
5. If the entered password matches the retrieved password, authentication is successful.
6. User is granted access to the system and redirected to the dashboard or designated landing page.
7. If the entered password does not match the retrieved password, authentication fails.
8. User is prompted with an error message indicating that the password is incorrect.
9. User can retry entering the password or utilize the "Forgot Password" feature.

Post conditions:

If authentication is successful, the user gains access to the system.
If authentication fails, the user is prompted to retry entering the password or reset the password using the "Forgot Password" feature.

Alternative Flow: Reset Password via Security Questions

Preconditions: User has initiated the login process.
User has forgotten their password.
User has set up security questions for password recovery.

Alternative Flow:

1. User clicks on the "Forgot Password" link.
2. System prompts the user to choose the option for password recovery.
3. User enters a new password and confirms it.
4. System updates the user's password in the database.
5. User receives confirmation that the password has been successfully reset.

Post conditions: User successfully resets their password and can now use the new password to log in.

Use case ID: UC1.1.4	**Use Case:** Display Login Error Message

Primary Actors: User

Preconditions: User attempts to log in to the system.
User provides incorrect login credentials (username and/or password).

Main Flow:

1. User submits login credentials (username and password).
2. System verifies the provided credentials against the database.
3. If the credentials are incorrect, the system displays an error message indicating the login failure.
4. The error message provides feedback to the user, informing them that the login attempt was unsuccessful due to incorrect credentials.
5. User is prompted to try logging in again or reset their password if needed.

Post conditions:

User receives feedback about the unsuccessful login attempt through the displayed error message.

Alternative Flow: Account Lockout

Preconditions: User attempts to log in to the system.
User provides incorrect login credentials (username and/or password) multiple times, exceeding the maximum allowed attempts.

Alternative Flow:

1. User submits login credentials (username and password).
2. System verifies the provided credentials against the database.
3. If the credentials are incorrect:
   a. The system tracks the number of failed login attempts for the user.
   b. If the number of failed attempts exceeds the maximum allowed threshold:
   i. The system locks the user's account temporarily to prevent further login attempts.
   ii. User receives a notification informing them that their account has been locked for security reasons.
   iii. User is prompted to contact customer support or follow a password reset process to regain access to their account.
   iv. User must follow the account recovery process to unlock their account.

Post conditions: User's account is temporarily locked due to multiple failed login attempts.

Use case ID: UC1.2	**Use Case:** Sign Up

Primary Actors: User

Preconditions: User accesses the sign-up page or interface of the system.
User has not already registered an account with the system.

Main Flow:

1. User navigates to the sign-up page or interface.
2. User provides required information such as username, email address, password, etc.
3. System validates the provided information:
   a. Checks if the username is unique and meets any specified requirements.
   b. Verifies the format and validity of the email address.
   c. Ensures the password meets the security criteria (e.g., minimum length, special characters).
4. If the information provided is valid:
   a. System creates a new user account with the provided details.
   b. User receives a confirmation message indicating successful registration.
   c. User is automatically logged into the system.
5. If the information provided is invalid:
   a. System displays error messages indicating the specific validation errors.
   b. User is prompted to correct the errors and resubmit the form.

Post conditions:
User successfully registers a new account with the system.

Alternative Flow: Password Requirements Not Met

Preconditions: User accesses the sign-up page or interface of the system.
User attempts to register an account with a password that does not meet the system's requirements.

Alternative Flow:

1. User navigates to the sign-up page or interface.
2. User provides the required information, including a username, email address, and password.
3. System validates the provided information.
4. System detects that the password provided by the user does not meet the system's requirements (e.g., minimum length, special characters).
5. System prompts the user with an error message indicating that the password requirements are not met.
6. User is informed about the specific password requirements that need to be fulfilled.
7. User revises the password to meet the specified requirements.
8. User resubmits the sign-up form with the revised password.
9. System validates the revised password and proceeds with the main flow of the sign-up process.

Post conditions: User successfully completes the sign-up process with a password that meets the system's requirements.

Use case ID: UC1.2.1	**Use Case:** Enter Name & Surname
Primary Actors: User	
Preconditions: User accesses the sign-up page or interface of the system.	
Main Flow: 1. User navigates to the sign-up page or interface. 2. User provides their name and surname in the designated fields. 3. System validates the provided information. 4. User proceeds to enter additional required information for sign-up (e.g., email address, password). 5. User completes the sign-up process by submitting the form.	
Post conditions: User successfully completes the sign-up process. User's name and surname are stored in the system database. User is registered as a new account holder in the system. User is logged into the system with their newly created account.	
Alternative Flow: Invalid Name & Surname Format	
Preconditions: User accesses the sign-up page or interface of the system.	
Alternative Flow: 1. User navigates to the sign-up page or interface. 2. User provides their name and surname in the designated fields. 3. System detects that the format of the provided name and/or surname is invalid (e.g., contains special characters, exceeds character limit). 4. System displays an error message prompting the user to correct the format of their name and/or surname. 5. User modifies the name and/or surname to meet the required format. 6. User proceeds to enter additional required information for sign-up (e.g., email address, password). 7. User completes the sign-up process by submitting the form.	
Post conditions: User successfully completes the sign-up process. User's name and surname, in the correct format, are stored in the system database. User is registered as a new account holder in the system. User is logged into the system with their newly created account.	

Use case ID: UC1.2.2	**Use Case:** Enter E-mail Address
Primary Actors: User	
Preconditions: User accesses the sign-up page or interface of the system.	
Main Flow: 1. User navigates to the sign-up page or interface. 2. User provides their email address in the designated field. 3. System validates the format of the email address provided by the user. 4. If the email address format is valid, the user proceeds to enter additional required information for sign-up (e.g., name, password). 5. If the email address format is invalid, the system displays an error message prompting the user to enter a valid email address. 6. User corrects the email address format if necessary. 7. User completes the sign-up process by submitting the form.	
Post conditions: User successfully completes the sign-up process. User's email address is stored in the system database. User is registered as a new account holder in the system. User is logged into the system with their newly created account.	
Alternative Flow: Error in Email Address Format	
Preconditions: User attempts to sign up for a new account. User accesses the sign-up page or interface.	
Alternative Flow: 1. User enters an email address in the designated field. 2. System attempts to validate the format of the email address provided by the user. 3. System detects that the email address format is invalid. 4. System prompts the user with an error message indicating the invalid email address format. 5. User is given the option to correct the email address format. 6. User revises the email address to comply with the required format. 7. User proceeds with the sign-up process by resubmitting the form.	
Post conditions: User successfully corrects the email address format. User completes the sign-up process with a valid email address. User's corrected email address is stored in the system database. User is registered as a new account holder in the system. User is logged into the system with their newly created account.	

Use case ID: UC1.2.3	**Use Case:** Create Password

Primary Actors: User

Preconditions: User attempts to sign up for a new account.
User has entered a valid email address and other required information.

Main Flow:

1. User navigates to the sign-up page or interface.
2. User fills out the sign-up form, including entering a password in the designated field.
3. System verifies the strength and validity of the password entered by the user.
4. System accepts the password if it meets the specified criteria (e.g., minimum length, combination of characters).
5. User submits the sign-up form with the entered password.

Post conditions:
User successfully creates a password for their new account.
User's password is securely stored in the system database.
User's account is registered and activated in the system.
User receives a confirmation message or email indicating successful sign-up.
User can now log in to the system using their email address and the newly created password.

Alternative Flow: Weak Password

Preconditions: User is attempting to sign up for a new account.
User has entered a password that does not meet the system's criteria for strength.

Alternative Flow:

1. User fills out the sign-up form, including entering a password that does not meet the system's password strength requirements.
2. System detects that the entered password is weak or does not meet the specified criteria.
3. System prompts the user with an error message indicating that the password is not strong enough.
4. User modifies the password to meet the system's password strength requirements.
5. User resubmits the sign-up form with the updated password.

Post conditions: User successfully creates a password that meets the system's password strength requirements.
User's password is securely stored in the system database.
User's account is registered and activated in the system.
User receives a confirmation message or email indicating successful sign-up.
User can now log in to the system using their email address and the newly created password.

Use case ID: UC02	**Use Case:** Entering Expenses and Incomes
Primary Actors: User	
Preconditions: System is operational. User must have access to the expense and income entry feature.	
Main Flow: 1. User enters expense or income details. 2. System checks and validates data. 3. User confirms entry. 4. System confirms entry.	
Post conditions: 1. User's financial records are updated. 2. System confirms entry.	
Alternative Flow: Data Error	
Preconditions: Errors detected during data validation.	
Alternative Flow: 1. System prompts the user to correct errors. 2. User makes corrections and resubmits. 3. Use case continues from main flow.	
Post conditions: 1. Corrected data accepted.	

Use case ID: UC2.1	**Use Case:** Choose Date
Primary Actors: User	
Preconditions: The user should have navigated to the screen or section where they can choose the date for entering expenses or income.	
Main Flow: 1. The user taps or clicks on the date selection field. 2. The system displays a date picker interface, allowing the user to select a date. 3. The user chooses a specific date from the date picker interface. 4. The selected date is confirmed or applied for entering expenses or income.	
Post conditions: The selected date is associated with the entry of expenses or income within the application.	
Alternative Flow: Cancel Date Selection	
Preconditions: The system must be functioning correctly without any technical issues.	
Alternative Flow: 1. The user decides to cancel the process of choosing a date. 2. The system returns the user to the main entry interface without selecting a date.	
Post conditions: No date is associated with the entry of expenses or income.	

Use case ID: UC2.2	**Use Case:** Enter Expense
Primary Actors: User	
Preconditions: The user must be authenticated and logged into the system to access the functionality for entering expenses.	
Main Flow: 1. The user initiates the process of entering expenses by accessing the entry interface. 2. The user inputs details such as the amount, category, date, and any additional notes for the expense entry. 3. The user confirms the entered details for the expense. 4. The user saves or submits the entered expenses, depending on the system's workflow.	
Post conditions: The system provides feedback confirming the successful recording of the expense entry..	
Alternative Flow: Invalid Expense Details	
Preconditions: User has initiated the process of entering expenses. Access to the entry interface is available. Stable system operation.	
Alternative Flow: 1. User inputs invalid or incomplete expense details. 2. System detects the invalid input and prompts the user to correct the errors. 3. User makes corrections to the expense details. 4. User resubmits the corrected expense details.	
Post conditions: System displays an error message indicating the invalid or incomplete expense details.	

Use case ID: UC2.2.1	**Use Case:** Choose Color
Primary Actors: User	
Preconditions: The system must be functioning correctly without any technical issues.	
Main Flow: 1. The user initiates the process of choosing a color for entering expenses by accessing the color selection interface. 2. The user selects a color from the available options provided by the system. 3. The user confirms the selected color for entering expenses. 4. The user saves or applies the selected color for entering expenses, depending on the system's workflow.	
Post conditions: The chosen color is successfully applied for entering expenses throughout the application interface	
Alternative Flow: Undo Color Selection	
Preconditions: User has initiated the process of choosing a color all expenses. Access to color selection interface is available. Stable system operation.	
Alternative Flow: 1. User selects a color but decides to undo the selection. 2. System allows the user to revert the color selection to the default or previous state.	
Post conditions: The previously selected color is removed, and the default or previous color state is restored.	

Use case ID: UC2.2.2	**Use Case:** Add Notes
Primary Actors: User	
Preconditions: The user should have navigated to the appropriate section or screen within the application where they can add notes for entering expenses.	
Main Flow: 1. The user initiates the process of adding notes for entering expenses by accessing the expense entry interface. 2. The user enters relevant notes or comments associated with the expense entry. 3. The user saves the entered notes to be associated with the expense entry. 4. The system provides feedback confirming that the notes have been successfully saved.	
Post conditions: The system confirms to the user that the notes have been saved successfully. The saved notes are displayed along with the expense entry for future reference.	
Alternative Flow: Edit Existing Notes	
Preconditions: User has initiated the process of adding notes for entering expenses. Access to expense entry interface with existing notes is available. Stable system operation.	
Alternative Flow: 1. User selects the option to edit existing notes associated with the expense entry. 2. System retrieves the existing notes for editing. 3. User modifies or adds additional notes. 4. User saves the edited or additional notes.	
Post conditions: The existing notes associated with the expense entry are successfully edited or augmented with additional content.	

Use case ID: UC2.3	**Use Case:** Enter Income
Primary Actors: User	
Preconditions: The user must be authenticated and logged into the system to access the functionality for entering income.	
Main Flow: 1. The user initiates the process of entering income by accessing the entry interface. 2. The user inputs details such as the amount, category, date, and any additional notes for the income entry. 3. The user confirms the entered details for the income. 4. The user saves or submits the entered income, depending on the system's workflow.	
Post conditions: The system provides feedback confirming the successful recording of the income entry.	
Alternative Flow: Invalid Income Details	
Preconditions: User has initiated the process of entering income. Access to the entry interface is available. Stable system operation.	
Alternative Flow: 5. User inputs invalid or incomplete income details. 6. System detects the invalid input and prompts the user to correct the errors. 7. User makes corrections to the income details. 8. User resubmits the corrected income details.	
Post conditions: System displays an error message indicating the invalid or incomplete income details.	

Use case ID: UC2.3.1	**Use Case:** Add Notes
Primary Actors: User	
Preconditions: The user should have navigated to the appropriate section or screen within the application where they can add notes for entering incomes.	
Main Flow: 5. The user initiates the process of adding notes for entering incomes by accessing the incomes entry interface. 6. The user enters relevant notes or comments associated with the incomes entry. 7. The user saves the entered notes to be associated with the incomes entry. 8. The system provides feedback confirming that the notes have been successfully saved.	
Post conditions: The system confirms to the user that the notes have been saved successfully. The saved notes are displayed along with the incomes entry for future reference.	
Alternative Flow: Edit Existing Notes	
Preconditions: User has initiated the process of adding notes for entering incomes. Access to incomes entry interface with existing notes is available. Stable system operation.	
Alternative Flow: 5. User selects the option to edit existing notes associated with the incomes entry. 6. System retrieves the existing notes for editing. 7. User modifies or adds additional notes. 8. User saves the edited or additional notes.	
Post conditions: The existing notes associated with the incomes entry are successfully edited or augmented with additional content.	

Use case ID: UC2.4	**Use Case:** View All Expenses and Income
Primary Actors: User	
Preconditions: The user must be authenticated and logged into the system to access the functionality for viewing expenses and income.	
Main Flow: 1. The user initiates the process of viewing all expenses and income by accessing the view interface. 2. The system retrieves and displays all expenses and income entries stored in the database. 3. The user can navigate through the entries and apply filters or search criteria to narrow down the displayed entries, if applicable. 4. The user can select individual entries to view detailed information such as amount, category, date, and notes.	
Post conditions: All expenses and income entries are displayed to the user as per their access permissions.	
Alternative Flow: Filter by Category	
Preconditions: User has initiated the process of viewing all expenses and income. Access to the view interface is available. Stable system operation.	
Alternative Flow: 1. User selects a specific category filter option. 2. System retrieves and displays only expenses and income entries belonging to the selected category.	
Post conditions: Only expenses and income entries belonging to the selected category are displayed.	

Use case ID: UC03	**Use Case:** Adding Future Expenses
Primary Actors: User	
Preconditions: System is operational. User authorized to add expenses.	
Main Flow: 1. User selects option to add future expenses. 2. User inputs expense amount, date, and description. 3. System confirms receipt of future expense. 4. System provides feedback on successful addition. 5. User returns to main interface.	
Post conditions: 1. If all required fields are filled and there are no errors, the future expense is successfully added to the system.	
Alternative Flow: User chooses to cancel the addition of future expenses	
Preconditions: The user's role and permissions within the system should allow them to cancel future expenses.	
Alternative Flow: 1. System cancels the addition process. 2. User returns to the main interface. 3. Use case ends.	
Post conditions: Any entered data is discarded	

Use case ID: UC3.1	**Use Case:** Search Expense
Primary Actors: User	
Preconditions: The user should have navigated to the appropriate section or screen within the application where they can perform expense searches.	
Main Flow: 1. The user initiates the process of searching expenses by accessing the search interface. 2. The user enters specific criteria, such as keywords, dates, categories, or amounts, to filter and search for expenses. 3. The system processes the entered search criteria and retrieves relevant expense records based on the provided filters. 4. The system presents the search results, displaying the expenses that match the entered criteria.	
Post conditions: The user can interact with the displayed search results, such as reviewing individual expense details or performing further actions.	
Alternative Flow: Filtered Views	
Preconditions: User has initiated the process of searching expenses. Access to search interface is available. Stable system operation.	
Alternative Flow: 1. User selects a predefined filter or category from a list. 2. System applies the selected filter to the expense records. 3. System displays the expenses filtered by the selected category or filter.	
Post conditions: The system successfully retrieves and displays expense records filtered by the selected category or filter.	

Use case ID: UC3.2	**Use Case:** Add Expense
Primary Actors: User	
Preconditions: The user should have navigated to the appropriate section or screen within the application where they can add new expenses.	
Main Flow: 1. The user initiates the process of adding a new expense by accessing the expense entry interface. 2. The user enters relevant details for the new expense, such as the amount, category, date, and any additional notes. 3. The user submits the entered expense details to the system for processing. 4. The system validates the entered expense details and saves them to the database. 5. The system provides confirmation to the user that the expense has been successfully added.	
Post conditions: The system displays a confirmation message to the user indicating that the expense has been successfully added.	
Alternative Flow: Recurring Expense Setup	
Preconditions: User has initiated the process of adding a new expense. Access to the expense entry interface is available. Stable system operation.	
Alternative Flow: 1. User selects the option for setting up a recurring expense. 2. User specifies recurrence details, such as frequency, start date, and end date. 3. User enters the expense details for the first occurrence. 4. System automatically generates future instances of the expense based on the specified recurrence settings.	
Post conditions: The system successfully saves the initial expense details and sets up future instances of the expense according to the recurrence settings.	

Use case ID: UC3.2.1	**Use Case:** Enter Expense Name
Primary Actors: User	
Preconditions: The user should have navigated to the appropriate section or screen within the application where they can enter new expenses name.	
Main Flow: 1. The user initiates the process of adding a new expense by accessing the expense entry interface. 2. The user enters a name or description for the new expense. 3. The user submits the entered expense name to the system for processing. 4. The system validates the entered expense name and saves it temporarily, awaiting additional expense details.	
Post conditions: The system successfully saves the entered expense name temporarily, awaiting additional details.	
Alternative Flow: Auto-Suggestion	
Preconditions: User has initiated the process of adding a new expense. Access to the expense entry interface is available. Stable system operation.	
Alternative Flow: 1. User begins typing the expense name into the designated field. 2. User selects an auto-suggested expense name instead of typing the full name.	
Post conditions: The system successfully records the selected expense name for the new expense entry.	

Use case ID: UC3.2.2	**Use Case:** Enter Expense Amount
Primary Actors: User	
Preconditions: The user should have navigated to the appropriate section or screen within the application where they can enter new expenses amount.	
Main Flow: 5. The user initiates the process of adding a new expenses amount by accessing the expense entry interface. 6. The user enters an amount or description for the new expense. 7. The user submits the entered expense amount to the system for processing. 8. The system validates the entered expense amount and saves it temporarily, awaiting additional expense details.	
Post conditions: The system successfully saves the entered expense amount temporarily, awaiting additional details.	
Alternative Flow: Manual Entry	
Preconditions: User has initiated the process of adding a new expense amount. Access to the expense entry interface is available. Stable system operation.	
Alternative Flow: 1. User manually enters the expense amount into the designated field. 2. System validates the entered amount for accuracy and completeness. 3. User confirms the entered amount and submits the expense.	
Post conditions: The system successfully saves the entered expense amount to the database.	

Use case ID: UC3.2.3	**Use Case:** Choose Payment Period
Primary Actors: User	
Preconditions: The user should have navigated to the appropriate section or screen within the application where they can choose payment period.	
Main Flow: 1. The user initiates the process of adding a new future expense. 2. The user chooses the desired payment period for the future expense from available options (e.g., weekly, monthly, annually). 3. The user confirms the selected payment period. 4. The system validates the chosen payment period and proceeds to the next step in the expense entry process.	
Post conditions: The system successfully records the chosen payment period for the future expense, ensuring it is properly scheduled for future payments according to the selected period.	
Alternative Flow: One-time Payment	
Preconditions: User authentication is successful. Access to the expense entry interface is available. Stable system operation. User selects the option for a one-time payment.	
Alternative Flow: 1. User chooses the "One-time" option for the payment period. 2. System confirms the selection of a one-time payment. 3. User proceeds to the next step in the expense entry process.	
Post conditions: The system records the one-time payment period for the future expense.	

Use case ID: UC3.2.4	**Use Case:** Enter Renewal
Primary Actors: User	
Preconditions: The user should have navigated to the appropriate section or screen within the application where they can enter renewal.	
Main Flow: 1. The user initiates the process of adding a new future expense. 2. The user inputs the necessary details related to the renewal, such as renewal type, renewal date, and any additional information. 3. The user confirms the entered renewal details. 4. The system validates the entered renewal information and proceeds to the next step in the expense entry process.	
Post conditions: The system successfully records the entered renewal information for the future expense, ensuring it is properly scheduled for renewal according to the specified date and details.	
Alternative Flow: Reminder Setup	
Preconditions: User authentication is successful. Access to the expense entry interface is available. Stable system operation. User indicates the desire to set up a reminder for the renewal.	
Alternative Flow: 1. User selects the option to set up a reminder for the renewal. 2. System prompts the user to specify the timing and frequency of the reminder. 3. User provides the desired timing and frequency for the reminder. 4. System validates the reminder settings. 5. User confirms the reminder setup.	
Post conditions: The system successfully sets up a reminder for the renewal according to the user's specified timing and frequency.	

Use case ID: UC3.2.5	**Use Case:** Choose Start Date
Primary Actors: User	
Preconditions: The user should have navigated to the appropriate section or screen within the application where they can choose start date.	
Main Flow: 1. The user initiates the process of choosing the start date for the future expense. 2. The user selects or inputs the desired start date for the future expense. 3. The user confirms the selected start date. 4. The system validates the selected start date and proceeds to the next step in the expense entry process.	
Post conditions: The system successfully records the chosen start date for the future expense, ensuring it is accurately scheduled according to the user's preference.	
Alternative Flow: Invalid Start Date	
Preconditions: User authentication is successful. Access to the expense entry interface is available. Stable system operation.	
Alternative Flow: 1. User selects a start date for the future expense. 2. The selected start date falls outside the acceptable range (e.g., in the past). 3. System detects the invalid start date. 4. System prompts the user to select a valid start date.	
Post conditions: The system alerts the user about the invalid start date and provides guidance for selecting a valid start date.	

Use case ID: UC3.2.6	**Use Case:** Add Reminder
Primary Actors: User	
Preconditions: The user should have navigated to the appropriate section or screen within the application where they can add reminders for future expenses.	
Main Flow: 1. The user initiates the process of adding a reminder for a future expense. 2. The user selects the expense for which they want to set a reminder. 3. The user selects the option to add a reminder for the selected expense. 4. The user inputs or selects the details of the reminder, such as the reminder date, time, frequency, and any additional notes. 5. The user confirms the reminder details and saves the reminder.	
Post conditions: The system records the reminder details for the selected expense, ensuring that the user receives timely notifications for the upcoming expense.	
Alternative Flow: Duplicate Reminder	
Preconditions: Same as the main flow preconditions.	
Alternative Flow: 1. User attempts to add a reminder for an expense that already has an existing reminder set. 2. System detects the duplicate reminder and alerts the user about the existing reminder. 3. User confirms whether they want to overwrite the existing reminder or keep both reminders.	
Post conditions: If the user chooses to overwrite the existing reminder, the system updates the reminder details.	

Use case ID: UC3.2.7	**Use Case:** Add Notes
Primary Actors: User	
Preconditions: The user must be authenticated and logged into the system to access the functionality for adding notes.	
Main Flow: 1. The user initiates the process of adding notes for a future expense. 2. The user selects the expense for which they want to add notes. 3. The user inputs or writes the notes relevant to the selected expense. 4. The user confirms the notes entered and saves them for the selected expense.	
Post conditions: The system associates the entered notes with the selected expense, ensuring that the user can refer to them when needed.	
Alternative Flow: Cancel Adding Notes	
Preconditions: The user must be authenticated and logged into the system. The user should be in the section for adding notes. The system must be functioning correctly.	
Alternative Flow: 1. User decides to cancel adding notes for the selected expense. 2. System cancels the note entry process and returns the user to the previous screen or interface without saving any changes.	
Post conditions: No notes are added or associated with the selected expense.	

Use case ID: UC3.3	**Use Case:** Monitor Incoming Expenses
Primary Actors: User	
Preconditions: The user should have access to the section designated for monitoring incoming expenses.	
Main Flow: 1. User navigates to the "Monitor Incoming Expenses" section of the application. 2. The system retrieves and displays a list of incoming expenses that are scheduled for the future. 3. User reviews the list of incoming expenses, which includes details such as expense name, amount, due date, and any associated notes. 4. User may have options to sort, filter, or search the list of incoming expenses based on their preferences. 5. User can take actions such as editing, deleting, or adding notes to individual incoming expenses as needed. 6. User may have the option to mark an incoming expense as paid once it is settled.	
Post conditions: The user has successfully monitored the incoming expenses.	
Alternative Flow: Error in Loading Incoming Expenses	
Preconditions: Same as main flow preconditions.	
Alternative Flow: 1. User attempts to navigate to the "Monitor Incoming Expenses" section. 2. The system encounters an error while loading the list of incoming expenses due to technical issues or connectivity issues. 3. The system displays an error message notifying the user about the issue and advises them to try again later or contact support if the problem persists. 4. User may choose to retry loading the incoming expenses or navigate to another section of the application.	
Post conditions: The user is informed about the error in loading the incoming expenses.	

Use case ID: UC3.3.1	**Use Case:** Delete Expense
Primary Actors: User	
Preconditions: The user should have access to the section designated for managing expenses.	
Main Flow: 1. User navigates to the "Delete Expense" section of the application. 2. The system retrieves and displays a list of existing expenses that the user has permission to delete. 3. User selects the expense(s) they wish to delete from the list. 4. User confirms the deletion action. 5. The system removes the selected expenses from the database.	
Post conditions: The selected expenses are successfully deleted from the system.	
Alternative Flow: Error in Deleting Expense	
Preconditions: Same as main flow preconditions.	
Alternative Flow: 1. User attempts to delete an expense. 2. The system encounters an error while processing the deletion request due to technical issues or database errors. 3. The system displays an error message notifying the user about the issue and advises them to try again later or contact support if the problem persists. 4. User may choose to retry deleting the expense or navigate to another section of the application.	
Post conditions: The user is informed about the error in deleting the expense.	

Use case ID: UC3.3.2	**Use Case:** Change Details
Primary Actors: User	
Preconditions: User is logged into the system. User has accessed the monitoring expenses section. There exist expenses in the system that can be modified.	
Main Flow: 1. User selects the expense they wish to modify. 2. System displays the details of the selected expense, such as the amount, category, description, and date. 3. User makes changes to the desired details of the expense. 4. User confirms the changes. 5. System updates the expense details with the new information. 6. System notifies the user that the changes have been successfully applied.	
Post conditions: User is informed that the changes have been successfully saved.	
Alternative Flow: Error in Modifying Expense Details	
Preconditions: Same as main flow preconditions.	
Alternative Flow: 1. User selects the expense they wish to modify. 2. System displays the details of the selected expense. 3. User attempts to make changes to the expense details. 4. System detects an error in the modification process, such as invalid input or system malfunction. 5. System prompts the user to correct the error. 6. User makes corrections and resubmits the modification. 7. System successfully applies the corrected changes. 8. System notifies the user that the changes have been successfully saved.	
Post conditions: The selected expense details are updated with the corrected modifications made by the user.	

Use case ID: UC04	**Use Case:** Monitor Calendar
Primary Actors: User	
Preconditions: System is operational. The calendar feature must be available and accessible to the user without any technical issues.	
Main Flow: 1. User opens the calendar feature. 2. User sees scheduled events. 3. User moves through dates. 4. User selects event for details. 5. User may edit event. 6. User goes back to main interface.	
Post conditions: If any changes were made, the calendar view reflects them.	
Alternative Flow: Error in Monitoring Calendar	
Preconditions: The user attempts to access the calendar feature within the system.	
Alternative Flow: 1. The system encounters an error while trying to load the calendar. 2. System displays an error message indicating the issue with loading the calendar. 3. User may retry accessing the calendar or contact support for assistance.	
Post conditions: User decides whether to retry or seek assistance.	

Use case ID: UC4.1	**Use Case:** Select Date
Primary Actors: User	
Preconditions: The user should open the calendar page.	
Main Flow: 1. Login/Sign up the application. 2. Open calendar page. 3. Select a specific date to add / monitor details. 4. Save changes.	
Post conditions: 1. The user ends up selecting the date wanted to add expense, income or event. 2. The date of the action is being saved to the database.	

Alternative Flow: 1. The user decides not to selects a specific date. 2. The changes on the system is not saved to the database.
Preconditions: The system does not save a specific date for current action.
Alternative Flow: The user logs out before saving a date.
Post conditions: The system does not save a specific date for current action.

Use case ID: UC 4.1.1	**Use Case:** Monitor Income
Primary Actors: User	
Preconditions: 1. The user should select date on the calendar manually. 2. The user decides to enter an income.	
Main Flow: 1. The user selects calendar section. 2. The user selects a specific date. 3. The system opens a page called “Enter event” 4. The user chooses to add income. 5. The user enters an amount incoming. 6. The user saves the changes via Save button.	
Post conditions: 1. The income amount is being saved to the database of the system to monitor in the future.	
Alternative Flow: 1. The user decides not to enter an income amount. 2. The user does not save changes. 3. The user does not want to monitor balance. 4. The user enters an invalid set of characters rather than entering an income.	
Preconditions: 1. The system does not save the income amount.	

2. The system shows an error massage to acknowledge the user.
Alternative Flow: None.
Post conditions: None.

Use case ID: UC 4.1.2	**Use Case:** Monitor Expense
Primary Actors: User	
Preconditions: 1. The user should select date on the calendar manually. 2. The user decides to enter an expense.	
Main Flow: 1. The user selects calendar section. 2. The user selects a specific date. 3. The system opens a page called "Enter event" 4. The user chooses to add expense. 5. The user enters an amount incoming. 6. The user saves the changes via Save button.	
Post conditions: 1. The expense amount is being saved to the database of the system to monitor in the future.	
Alternative Flow: 1. The user decides not to enter an expense amount. 2. The user does not save changes. 3. The user does not want to monitor balance. 4. The user enters an invalid set of characters rather than entering an expense.	
Preconditions: 1. The system does not save the expense amount. 2. The system shows an error massage to acknowledge the user.	
Alternative Flow: None.	
Post conditions: None.	

Use case ID: UC 4.1.3	**Use Case:** Monitor Schedule
Primary Actors: User	
Preconditions: 1. The user should select date on the calendar manually. 2. The user decides to enter an expense.	
Main Flow: 1. The user selects calendar section. 2. The user selects a specific date. 3. The system opens a page called “Enter event” 4. The user chooses to add event. 5. The user enters a duration. 6. The user saves the changes via Save button.	
Post conditions: 1. The event is being saved to the database of the system to monitor in the future	
Alternative Flow: 1. The user decides not to enter an event. 2. The user does not save changes. 3. The user does not want to monitor schedule. 4. The user enters an invalid set of characters rather than entering a string.	
Preconditions: 1. The system does not save the event time. 2. The system shows an error massage to acknowledge the user.	
Alternative Flow: None.	
Post conditions: None.	

Use case ID: UC4.2	**Use Case:** Add Event
Primary Actors: User	
Preconditions: User presses Add (+) button on the Calendar page.	

Main Flow:

5. User enters Username and Password
6. Presses Calendar section.
7. Presses Add (+) button on the right-below side on the page

Post conditions:

1. Show a point below the current or chosen date for acknowledging user an event is set.
2. Show recently added event on the schedule underneath the chosen date.

Alternative Flow: Cancel adding event process.

Preconditions: The user must choose to cancel the addition of the events before compelting the process.

Alternative Flow:

1. User chooses to cancel adding the event
2. System discard entered data.
3. Use case ends.

Post conditions:

1. The event entry process is terminated.
2. Any data entered by the user during the event creation process is discarded.

Use case ID: UC4.2.1	**Use Case:** Enter Event Name

Primary Actors: User

Preconditions: User presses Add (+) button on the Calendar page.

Main Flow:

1. User enters Username and Password
2. Presses Calendar section.
3. Presses Add (+) button on the right-below side on the page
4. Enter event name.
5. Press “Save” button.

Post conditions:

1. Show a point below the current or chosen date for acknowledging user an event is set.
2. Show recently added event on the schedule underneath the chosen date.

Alternative Flow:

1. User enters an event name that does not meet the system's validation criteria

2. User acknowledges the error message and attempts to correct the event name.
3. User resubmits the corrected event name.

Preconditions: The user should enter an invalid event name.

Alternative Flow:

1. User chooses to cancel adding the event
2. System discard entered data.
3. Use case ends.

Post conditions:

1. The event entry process is terminated.
2. Any data entered by the user during the event creation process is discarded.

Use case ID: UC 4.1.3	**Use Case:** Choose Date

Primary Actors: User

Preconditions:
1. Login / Sign up to the application
2. The user should open the calendar page.

Main Flow:

1. Login/Sign up the application.
2. Open calendar page.
3. Add event via + button.
4. Choose date on the Add Event Menu.
5. Save changes.

Post conditions:

1. The system saves an event with its specific date to the database system.

Alternative Flow:

1. The user does not choose to add event.
2. The user does not save the changes.

Preconditions: 1. The system does not save any date to the database.

Alternative Flow: None.

Post conditions: None.

Use case ID: UC4.2.3	**Use Case:** Enter Duration
Primary Actors: User	
Preconditions: 1. Login / Sign up to the application 2. The user should open the calendar page. 3. User adds event.	
Main Flow: 1. User enters Username and Password 2. Presses Calendar section. 3. Presses Add (+) button on the right-below side on the page 4. User enters an event name. 5. User chooses a time span.	
Post conditions: 1. User enters a correct event name.	
Alternative Flow: 1. User refuses adding duration of the event.	
Preconditions: The user prefers not to enter a duration.	
Alternative Flow: Continue adding event process.	
Post conditions: None.	

Use case ID: UC 4.2.5	**Use Case:** Add Reminder
Primary Actors: User	
Preconditions: 1. Login / Sign up to the application 2. The user should open the calendar page. 3. User adds event.	

Main Flow:

1. User enters Username and Password
2. Presses Calendar section.
3. Presses Add (+) button on the right-below side on the page
4. User enters an event name.
5. User decides to add reminder.
6. User determines the frequency of it.
7. Save changes.

Post conditions:

1. The system sends a reminder notification to the user via application on given date.

Alternative Flow:

1. The user decides not to fix a reminder.
2. The user does not save the changes.
3. The user logs out before save changes.

Preconditions: The user prefers not to enter a reminder.

Alternative Flow:

1. Continue adding event process.
2. Save changes.

Post conditions: The event can be monitored on the schedule section.

Use case ID: UC 4.2.6	**Use Case:** Enter Repeat Time
Primary Actors: User	

Preconditions:

1. Login / Sign up to the application
2. The user should open the calendar page.
3. User adds event.

Main Flow:

1. User enters Username and Password
2. Presses Calendar section.
3. Presses Add (+) button on the right-below side on the page
4. User enters an event name.
5. User decides to add reminder.
6. User determines the repetition frequency of it.
7. Save changes.

Post conditions:

1. The system repeats and shows on the calendar according to the repetition frequency of the event.

Alternative Flow:

1. The user decides not to fix a repetition frequency.
2. The user does not save the changes.
3. The user logs out before save changes.

Preconditions: The user prefers not to enter a repetition frequency.

Alternative Flow:

1. Continue adding event process.
2. Save changes.

Post conditions: The event can be monitored on the schedule section.

Use case ID: UC 4.2.7	**Use Case:** Choose Color
Primary Actors: User	

Preconditions:

1. Login / Sign up to the application
2. The user should open the calendar page.
3. User adds event.

Main Flow:

1. User enters Username and Password
2. Presses Calendar section.
3. Presses Add (+) button on the right-below side on the page
4. User enters an event name.
5. User determines the color of it.
6. Save changes.

Post conditions: 1. The system shows colors according to user preference on the schedule section.

Alternative Flow:

1. The user decides not to customize event.
2. The user does not save the changes.
3. The user logs out before save changes.

Preconditions: The user prefers not to customize the event.

Alternative Flow:

1. Continue adding event process.
2. Save changes.

Post conditions: The event and color of it can be monitored on the schedule section.

Use case ID: UC 4.2.8	**Use Case:** Add Note
Primary Actors: User	
Preconditions: 1. Login / Sign up to the application 2. The user should open the calendar page. 3. User adds event.	

Main Flow:

1. User enters Username and Password
2. Presses Calendar section.
3. Presses Add (+) button on the right-below side on the page
4. User enters an event name.
5. User determines whether add notes or not.
6. Save changes.

Post conditions: 1. The system shows notes of the event written by the user.

Alternative Flow:

1. The user decides not to add notes
2. The user does not save the changes.
3. The user logs out before save changes.

Preconditions: The user prefers not to add notes.

Alternative Flow:

1. Continue adding event process.
2. Save changes.

Post conditions: The adding event process is terminated.

Use case ID: UC05	**Use Case:** Open Hamburger Bar
Primary Actors: User	
Preconditions: The hamburger menu icon should be visible and accessible on the screen.	
Main Flow: 1. The user taps or clicks on the hamburger menu icon. 2. The system displays the expanded hamburger menu containing navigation options or additional features. 3. The user can select an option from the opened hamburger menu.	
Post conditions: After the user selects an option or interacts with the menu, the hamburger menu closes automatically or upon user action.	
Alternative Flow: Error in Opening Hamburger Bar	
Preconditions: The user attempts to interact with the hamburger bar.	
Alternative Flow: 1. The system encounters an error while attempting to open the hamburger menu. 2. System displays an error message indicating the issue with opening the hamburger menu. 3. User may retry opening the hamburger menu or close the error message.	
Post conditions: User is informed about the error encountered while opening the hamburger menu.	

Use case ID: UC5.1	**Use Case:** Click & Open the Bar
Primary Actors: User	
Preconditions: User is logged into the system. The hamburger menu bar icon is visible on the user interface. User intends to access additional features or navigation options.	
Main Flow: 1. User clicks on the hamburger menu bar icon. 2. System recognizes the user's action and opens the menu bar. 3. System displays a list of available options or features within the menu. 4. User selects a specific option or feature from the menu.	
Post conditions: The menu bar remains open until the user either selects an option or clicks outside the menu to close it.	
Alternative Flow: Error Opening the Bar	
Preconditions: User is logged into the system. The hamburger menu bar icon is visible on the user interface.	
Alternative Flow: 1. User attempts to click on the hamburger menu bar icon to open the menu. 2. System encounters an error due to technical issues or system overload. 3. System prompts the user with an error message indicating the failure to open the menu. 4. User can choose to retry opening the menu or dismiss the error message.	
Post conditions: If the user retries opening the menu, the system makes another attempt to open it.	

Use case ID: UC5.1.1	**Use Case:** Log Out
Primary Actors: User	
Preconditions: User is logged into the system. The hamburger menu bar icon is visible on the user interface.	
Main Flow: 1. User clicks on the hamburger menu bar icon to open the menu. 2. User selects the "Log Out" option from the menu. 3. System prompts the user to confirm the logout action. 4. User confirms the logout action. 5. System logs out the user and redirects them to the login page. 6. User is successfully logged out of the system.	
Post conditions: User is logged out of the system and redirected to the login page.	
Alternative Flow: Canceling Logout	
Preconditions: User is logged into the system. The hamburger menu bar icon is visible on the user interface.	
Alternative Flow: 1. User clicks on the hamburger menu bar icon to open the menu. 2. User selects the "Log Out" option from the menu. 3. System prompts the user to confirm the logout action. 4. Instead of confirming, the user decides to cancel the logout. 5. System cancels the logout action and returns the user to the previous screen.	
Post conditions: User session and authentication tokens remain active.	

Use case ID: UC5.1.2	**Use Case:** Contact
Primary Actors: User	
Preconditions: User is logged into the system. The hamburger menu bar icon is visible on the user interface. User has selected the "Contact" option from the hamburger menu.	
Main Flow: 1. User clicks on the hamburger menu bar icon to open the menu. 2. User selects the "Contact" option from the menu. 3. System displays the contact information screen. 4. User can view the contact details such as email address, phone number, or contact form. 5. User can choose to contact support via email, phone, or by filling out a contact form. 6. User inputs the necessary information for contacting support. 7. User submits the contact request.	
Post conditions: User receives confirmation of the contact submission.	
Alternative Flow: No Contact Information Provided	
Preconditions: User is logged into the system. The hamburger menu bar icon is visible on the user interface. User has selected the "Contact" option from the hamburger menu.	
Alternative Flow: 1. User clicks on the hamburger menu bar icon to open the menu. 2. User selects the "Contact" option from the menu. 3. System displays the contact information screen. 4. User notices that no contact information is available. 5. User decides not to proceed with contacting support due to lack of contact information. 6. User closes the contact screen.	
Post conditions: User remains on the same screen without contacting support.	

Use case ID: UC5.1.3	**Use Case:** Click & Open Settings
Primary Actors: User	
Preconditions: User is logged into the system. The hamburger menu bar icon is visible on the user interface.	
Main Flow: 1. User clicks on the hamburger menu bar icon to open the menu. 2. User selects the "Settings" option from the menu. 3. System navigates to the settings screen. 4. User locates the option to change their username. 5. User enters the new name in the designated field. 6. User saves the changes and exits the settings screen.	
Post conditions: User's name is updated with the new changes.	
Alternative Flow: Cancel Name Change	
Preconditions: User is logged into the system. The hamburger menu bar icon is visible on the user interface. User is on the settings screen.	
Alternative Flow: 1. User clicks on the option to change their name. 2. User enters the new name but decides to cancel the change. 3. User selects the "Cancel" or "Back" button. 4. System discards the changes made to the name field. 5. User is returned to the settings screen without any modifications.	
Post conditions: No changes are made to the user's name.	

Use case ID: UC5.2	**Use Case:** Click & Close the Bar
Primary Actors: User	
Preconditions: The hamburger bar is currently open on the user interface.	
Main Flow: 1. User clicks on the close button (e.g., "X" icon) located on the hamburger bar. 2. System detects the user's action and closes the hamburger bar. 3. User interface returns to its original state without the hamburger menu visible.	
Post conditions: The hamburger bar is successfully closed.	
Alternative Flow: Hamburger Bar Already Closed	
Preconditions: The hamburger bar is not currently visible on the user interface.	
Alternative Flow: 1. User attempts to close the hamburger bar, but it is already closed. 2. System ignores the action as the hamburger bar is already in a closed state. 3. User interface remains unchanged.	
Post conditions: No changes occur as the hamburger bar was already closed.	

2.6. NON-FUNCTIONAL REQUIREMENTS

- The system shall load in 10 seconds, be responsive, and have few lags or crashes.
- The system shall operate in 15 secs. and efficiently, minimizing user wait times.
- The system shall ensure the security of user data.
- The application shall prevent unauthorized access by limiting the number of login attempts.
- The system shall save a user-friendly interface that allows users to navigate comfortably, and perform tasks.
- The system shall be resistant to increased user numbers and data loads, maintaining performance as it expands.
- The system shall cover elements like speed, scalability, and resource usage.
- The system shall be simple to maintain and upgrade.
- The system shall have an availability rating of at least 99.9%, as measured by monitoring system up time over a period of one month.
- The system shall provide working on different mobile platforms (iOS, Android) and compatibility with various devices.
- The system shall provide algorithms for backing up user data and restoring if it needed.
- The system shall uphold the privacy of user data, compliance with GDPR (General Data Protection Regulation) and other data protection laws.
- The system shall contain elements like authentication and authorization.
- The system shall provide the amount of storage it needs.
- The application shall have a usability rating of at least 80%, as measured by user testing conducted with a representative sample of users.
- The application shall have a Mean Time Between Critical Failures (MTBCF) of at least 30 days, as measured by tracking the time between critical failures over a period of one year.
- The system shall have portability requirements addressing the ease with which software can be transferred from one environment to another, such as different operating systems or devices.
- The system shall have availability requirements focused on ensuring that the system is accessible to users with minimal downtime due to failures.

2.7. VOLERE TEMPLATE

Requirement ID: 01	**Requirement Type**: NFR (Performance)	**Event/Use case # 1 - 3**
Description: he system shall load in 10 seconds, be responsive, and have few lags or crashes.		
Rationale: It enhances user experience, minimizes work flow disruptions, and reduces the risk of data loss, leading to increased productivity and user satisfaction.		
Fit Criteria: After at least 1 million users upload and start to use, loading speed should be still fast, responsiveness should be high, and occurrences of lags or crashes should be minimal.		
Priority: Necessary		

Requirement ID: 02	**Requirement Type**: NFR (Performance)	**Event/Use case # 1**
Description: The system shall operate in 15 seconds, minimizing user wait times.		
Rationale: It enhances customer satisfaction, optimizing operational efficiency, and improving overall service quality.		
Fit Criteria: After at least 1 million users upload and start to use, the system should operate with speed and efficiency, ensuring minimal user wait times.		
Priority: Necessary		

Requirement ID: 03	**Requirement Type**: NFR (Security)	**Event/Use case # 1 - 2**
Description: The system shall ensure the security of user data.		
Rationale: Users want to see that the app has deeply rooted in safeguarding sensitive information, adhering to regulatory requirements, and establishing trust with users.		
Fit Criteria: Making all software tests for protect information and security.		
Priority: Necessary		

Requirement ID: 04	**Requirement Type**: NFR (Performance)	**Event/Use case # 1**
Description: The application shall prevent unauthorized access by limiting the number of login attempts.		
Rationale: This security measure can also help organizations comply with industry regulations and standards for data protection, reducing the risk of legal consequences and reputation damage in the event of a breach.		
Fit Criteria: Complete to cybersecurity tests and analysis for protect to limited number of wrong logins.		
Priority: Necessary		

Requirement ID: 05	**Requirement Type**: NFR (Performance)	**Event/Use case # 3 - 4 - 5**
Description: The system shall saving a user-friendly interface that allows users to navigate comfortably and perform tasks.		
Rationale: It enhances the overall user experience, promoting ease of navigation, ensuring accessibility, improving efficiency, reducing the learning curve, and fostering a culture of continuous improvement based on user feedback.		
Fit Criteria: Feedback from the inside of team and users after the systematic preparation of application content and usage.		
Priority: Desirable		

Requirement ID: 06	**Requirement Type**: NFR (Performance)	**Event/Use case # 2**
Description: The system shall be resistant to increased user numbers and data loads, maintaining performance as it expands.		
Rationale: It roots in scalability, user experience optimization, business continuity, future-proofing, data integrity and security, and gaining a competitive advantage in the market.		
Fit Criteria: The software and systems team conduct robust testing to test application durability and data performance.		
Priority: Desirable		

Requirement ID: 07	**Requirement Type**: NFR (Performance)	**Event/Use case # 3 - 5**
Description: The system shall cover elements like speed, scalability, and resource usage.		
Rationale: Uses resources efficiently, organizations can not only provide a better user experience but also maximize their return on investment, minimize downtime, and avoid the need for costly hardware upgrades.		
Fit Criteria: Testing system operation and speed by ensuring that application usage is done simultaneously by a large number of users.		
Priority: Desirable		

Requirement ID: 08	**Requirement Type**: NFR (Performance)	**Event/Use case # 1- 2**
Description: The system shall be simple to maintain and upgrade.		
Rationale: It caused by in promoting agility, reducing downtime, improving reliability and security, enhancing user satisfaction, and achieving cost-efficiency.		
Fit Criteria: The software team conducts continuous testing in background and fixes in-application errors through updates.		
Priority: Desirable		

Requirement ID: 09	**Requirement Type**: NFR (Reliability)	**Event/Use case # 2- 3 - 4**
Description: The system shall have an availability rating of at least 99.9%, as measured by monitoring system up time over a period of one month.		
Rationale: Measuring the system up time over a period of one month provides an accurate assessment of the application's reliability and its ability to maintain high availability.		
Fit Criteria: Ensuring that the application remains operational by performing background system tests by the team at regular intervals.		
Priority: Necessary		

Requirement ID: 10	**Requirement Type**: NFR (Platform)	**Event/Use case # 1 - 2**
Description: The system shall provide working on different mobile platforms (iOS, Android) and compatibility with various devices.		

Rationale: The rationale for providing compatibility with different mobile platforms (iOS, Android) and various devices is rooted in expanding market reach, improving user convenience and accessibility, ensuring adaptability to market trends, maintaining user experience consistency, and facilitating ecosystem integration.
Fit Criteria: Taking feedback from stakeholders and work with them to manage background app management.
Priority: Valuable

Requirement ID: 11	**Requirement Type**: NFR (Reliability)	**Event/Use case # 1**
Description: The system shall provide algorithms for backing up user data and restoring if it needed.		
Rationale: It caused by in data protection, disaster recovery preparedness, data integrity maintenance, user trust and confidence, legal compliance, operational efficiency, and system continuity planning.		
Fit Criteria: Performing routine software tests and developments to perform system backup and recovery operations when it needed.		
Priority: Necessary		

Requirement ID: 12	**Requirement Type**: NFR (Security)	**Event/Use case # 1- 2- 3**
Description: The system shall uphold the privacy of user data, compliance with GDPR (General Data Protection Regulation) and other data protection laws.		
Rationale: It encompasses maintaining user trust and confidence, ensuring legal compliance, enhancing data security, respecting user rights and consent, facilitating cross-border data transfers, and building a positive reputation and brand image.		
Fit Criteria: Making continuous performance of systemic firewalls and software protection tests for application and data security.		
Priority: Desirable		

Requirement ID: 13	**Requirement Type**: NFR (Security)	**Event/Use case # 1**
Description: The system shall be contains elements like authentication and authorization.		

Rationale: It can help organizations comply with industry regulations and standards for data protection, reducing the risk of legal consequences and reputation damage in the event of a breach.
Fit Criteria: Ensuring ongoing testing and protection for user security and cybersecurity.
Priority: Desirable

Requirement ID: 14	**Requirement Type**: NFR (Reliability)	**Event/Use case # 4**
Description: The system shall provide to the amount of storage it needs.		
Rationale: By providing the system to dynamically allocate storage resources based on actual needs, the system can effectively manage data growth, optimize resource usage, and maintain high performance and reliability over time.		
Fit Criteria: Difference between capacity of cloud and current storage.		
Priority: Necessary		

Requirement ID: 15	**Requirement Type**: NFR (Usability)	**Event/Use case # 2 - 3**
Description: The application shall have a usability rating of at least 80%, as measured by user testing conducted with a representative sample of users.		
Rationale: By tailoring the application's settings to their individual preferences, users can have a more personalized experience with the application.		
Fit Criteria: Providing usage tests and user feedback for the comfort and continuity of application use.		
Priority: Desirable		

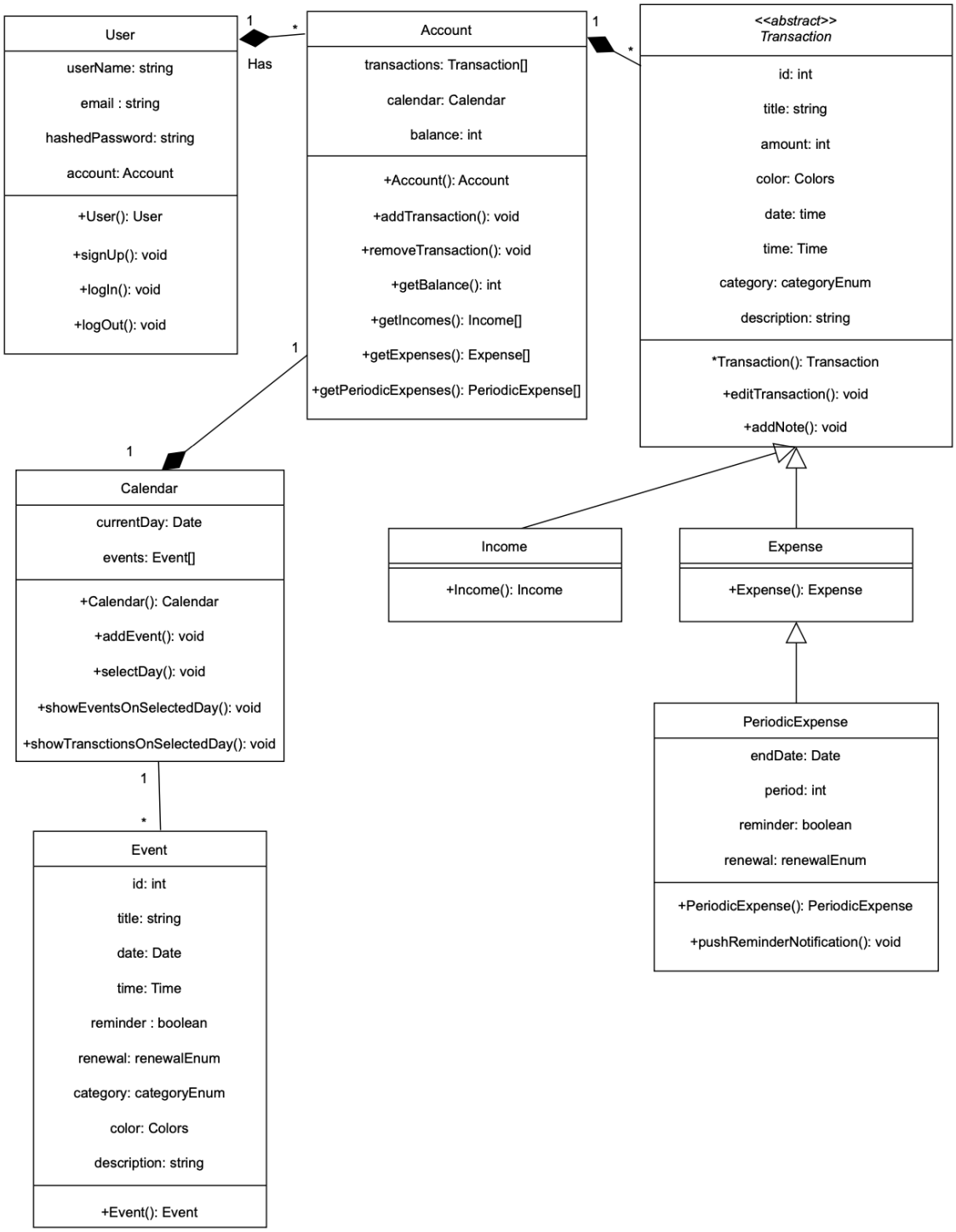
Requirement ID: 16	**Requirement Type**: NFR (Reliability)	**Event/Use case # 1-2-4**
Description: The application shall have a Mean Time Between Critical Failures (MTBCF) of at least 30 days, as measured by tracking the time between critical failures over a period of one year.		

Rationale: The application is reliable and minimizes downtime, which is crucial for user satisfaction.
Fit Criteria: Implementation of tests with the proper specifications.
Priority: Desirable

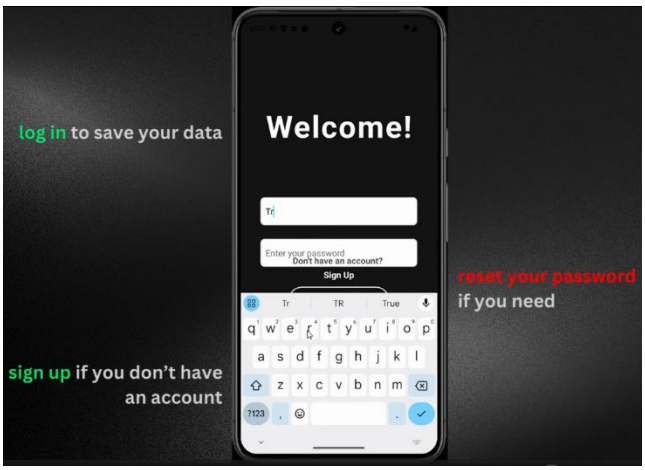
Requirement ID: 17	**Requirement Type**: NFR (Performance)	**Event/Use case #1-2-4**
Description: The system shall have portability requirements address the ease with which software can be transferred from one environment to another, such as different operating systems or devices.		
Rationale: The rationale for incorporating portability requirements into the system encompasses interoperability, user flexibility, market reach, technology adaptability, reduced migration efforts, cost efficiency, and compliance with standards.		
Fit Criteria: In order to avoid problems in data transfer related by device replacement, the software team works and makes tests according to the continuous feedback from the users.		
Priority: Desirable		

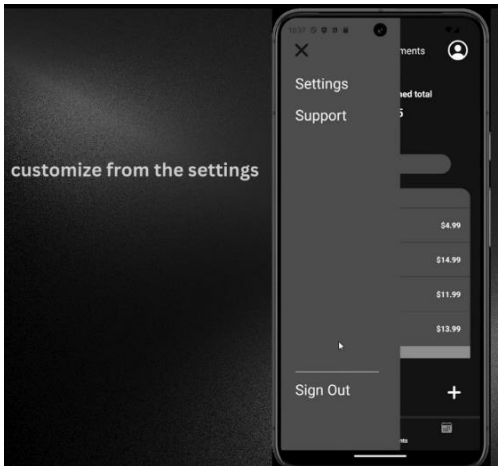
Requirement ID: 18	**Requirement Type**: NFR (Performance)	**Event/Use case # 1**
Description: The system shall have availability requirements focus on ensuring that the system is accessible to users with minimal downtime due to failures.		
Rationale: This requirement including availability requirements in the system focuses on enhancing the user experience, promoting customer satisfaction, improving operational efficiency, mitigating risks, and gaining a competitive advantage.		
Fit Criteria: Receiving feedback from users and conducting tests to ensure that system downtime is kept to a minimum and application usage continuity is ensured.		
Priority: Desirable		

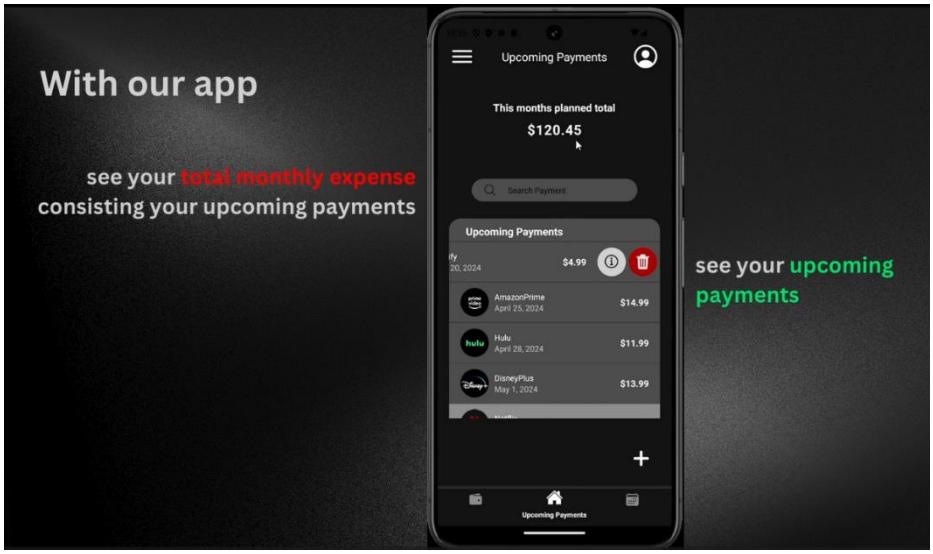
3. CLASS DIAGRAM

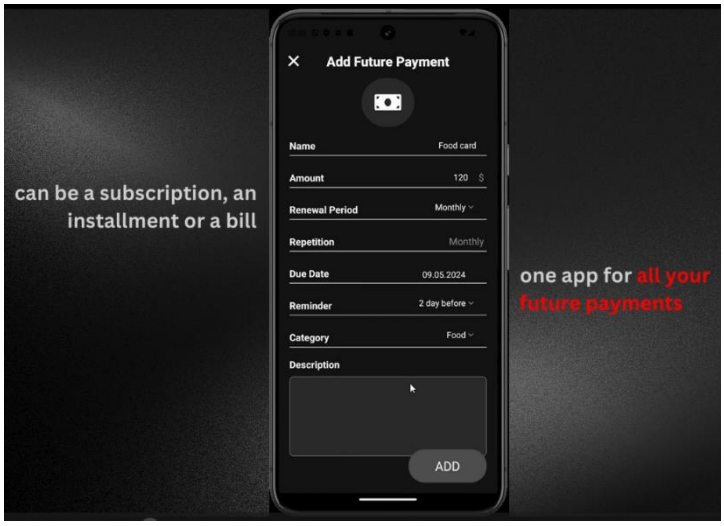


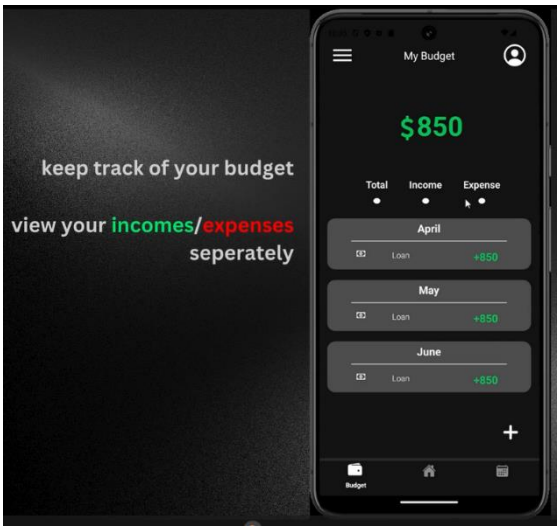
4. USER INTERFACE DESIGNS

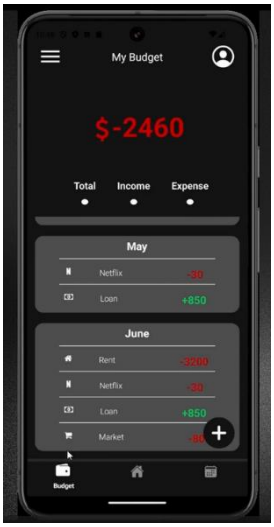


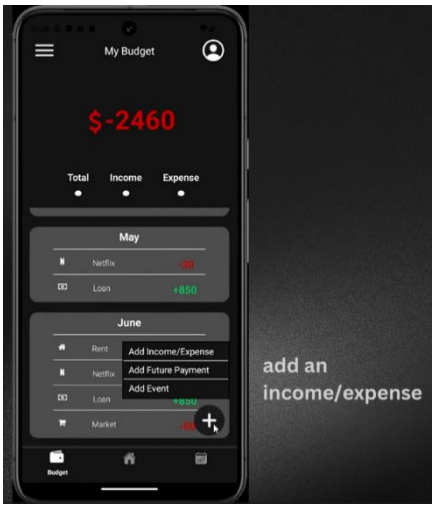


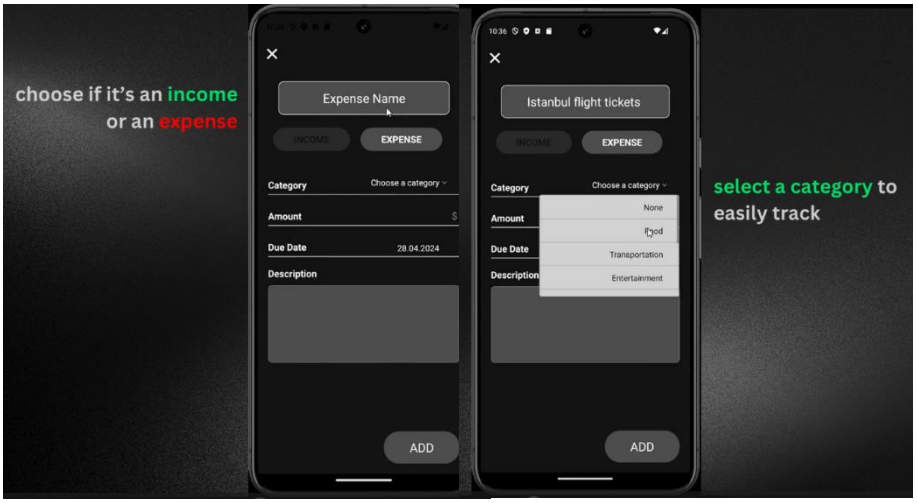


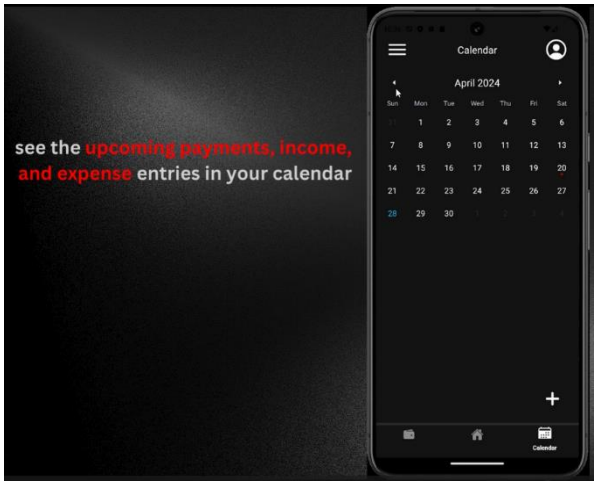


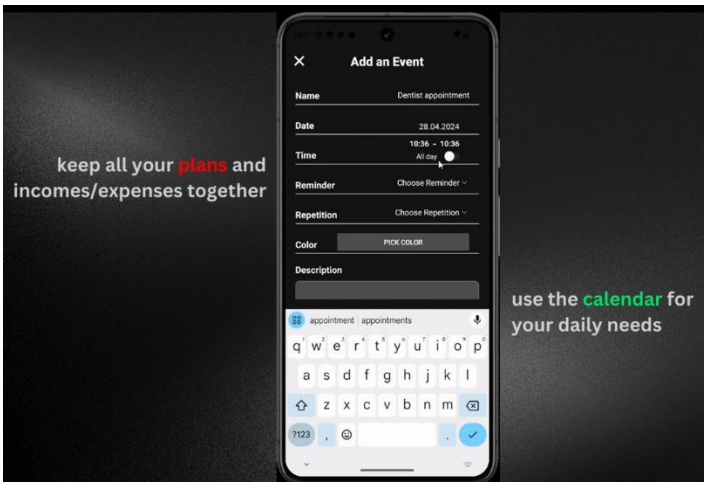

keep track of your budget
view your incomes/expenses seperately
My Budget
$850
Total
Income
Expense
April
Loan
+850
May
Loan
+850
June
Loan
+850
Budget


My Budget
$-2460
Total
Income
Expense
May
Netflix
Loan
+850
June
Rent
Netflix
Loan
+850
Market
Budget


My Budget
$-2460
Total
Income
Expense
May
Netflix
Loan
+850
June
Rent
Add Income/Expense
Netflix
Add Future Payment
Add Event
Loan
Market
Budget
add an income/expense


choose if it's an income or an expense
Expense Name
INCOME
EXPENSE
Category
Choose a category
Amount
Due Date
28.04.2024
Description
ADD
Istanbul flight tickets
INCOME
EXPENSE
Category
Choose a category
Amount
Due Date
Description
None
Food
Transportation
Entertainment
ADD
select a category to easily track


see the upcoming payments, income, and expense entries in your calendar
Calendar
April 2024
Sun
Mon
Tue
Wed
Thu
Fri
Sat
Calendar


keep all your plans and incomes/expenses together
Add an Event
Name
Dentist appointment
Date
28.04.2024
Time
10:36 – 10:36
All day
Reminder
Choose Reminder
Repetition
Choose Repetition
Color
PICK COLOR
Description
use the calendar for your daily needs

5. DETAILED CLASS DIAGRAM

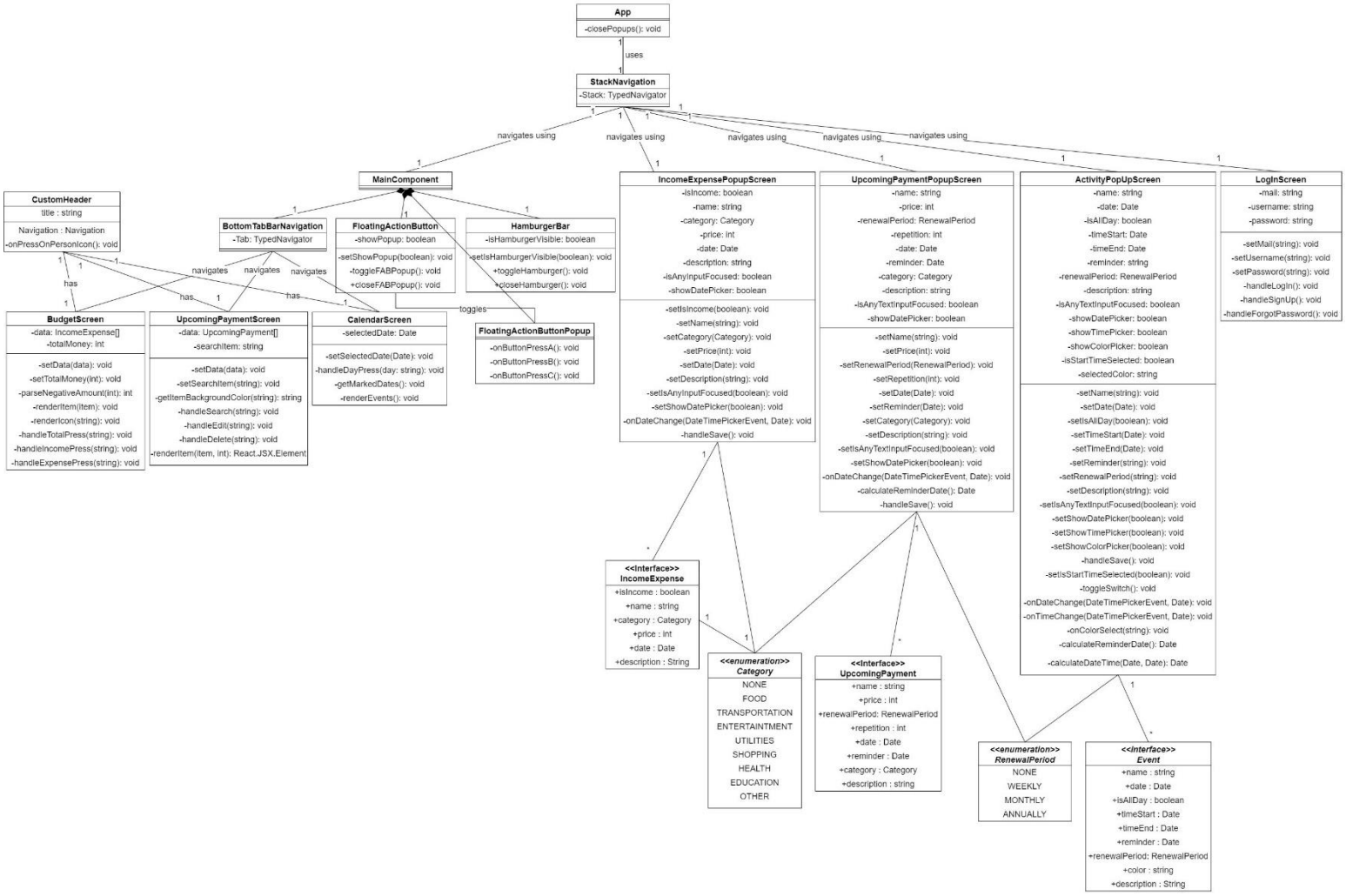


Fig. 1. *Detailed class diagram of the system (including associations, relationships etc.).*

6. DYNAMIC MODELS

6.1. SEQUENCE DIAGRAMS

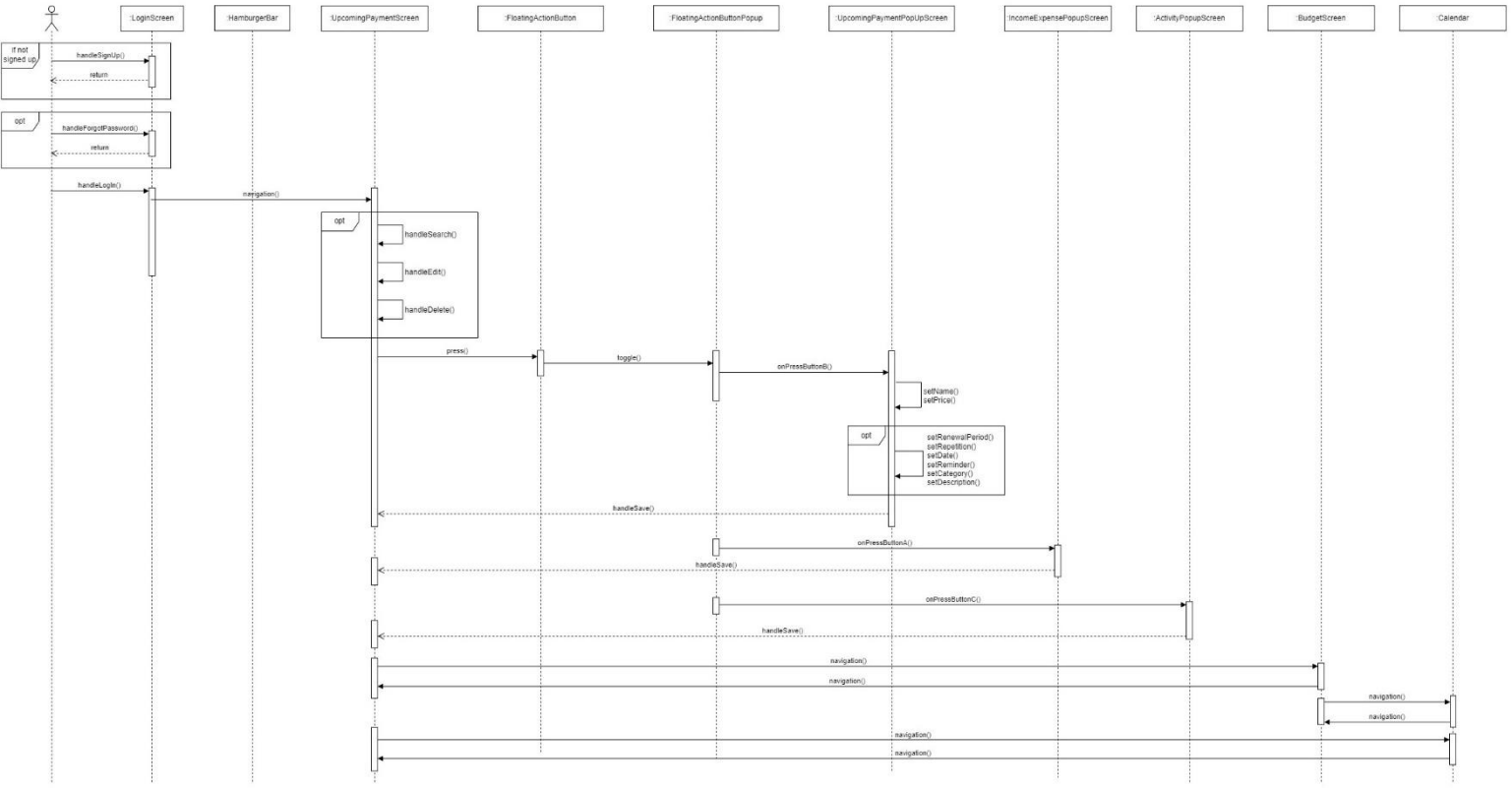


Fig. 2. *Sequence diagram shows the flow of the login screen (authentication), navigation between the screens, upcoming payment screen and upcoming payment pop up screen.*

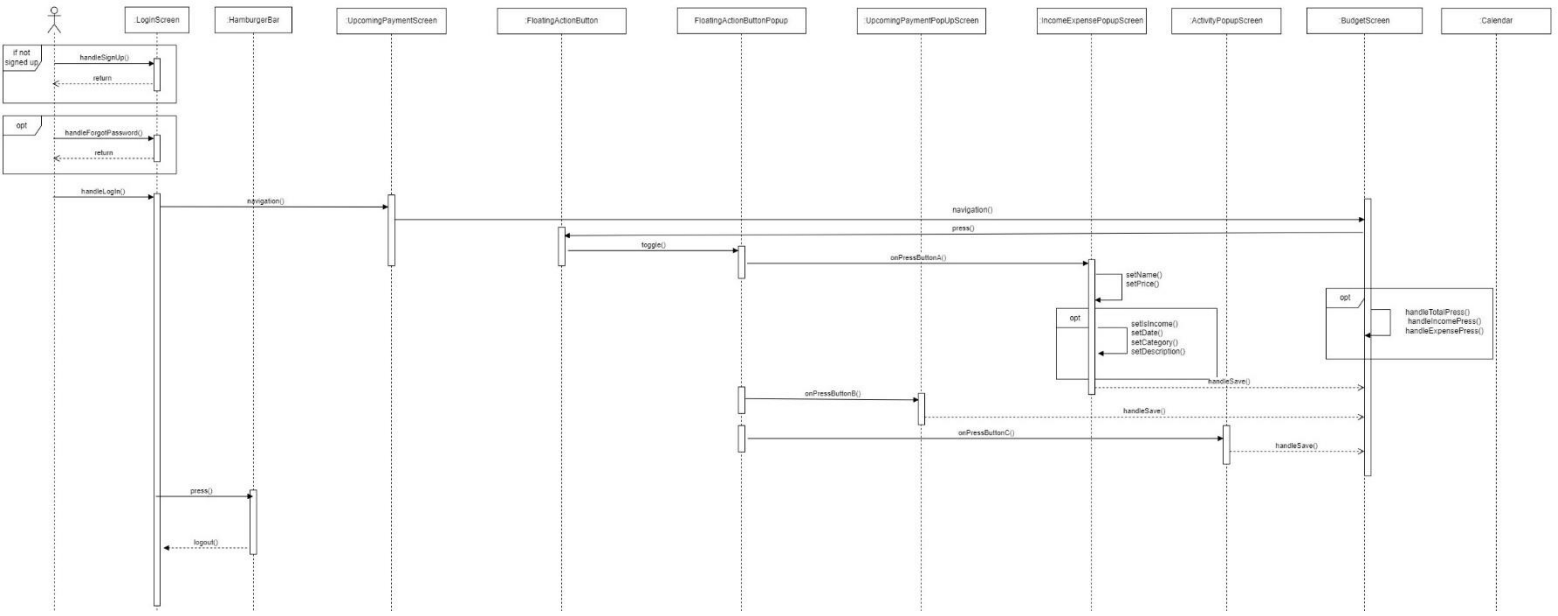


Fig. 3. *Sequence diagram shows the flow of the hamburger bar, budget screen and income-expense pop up screen.*

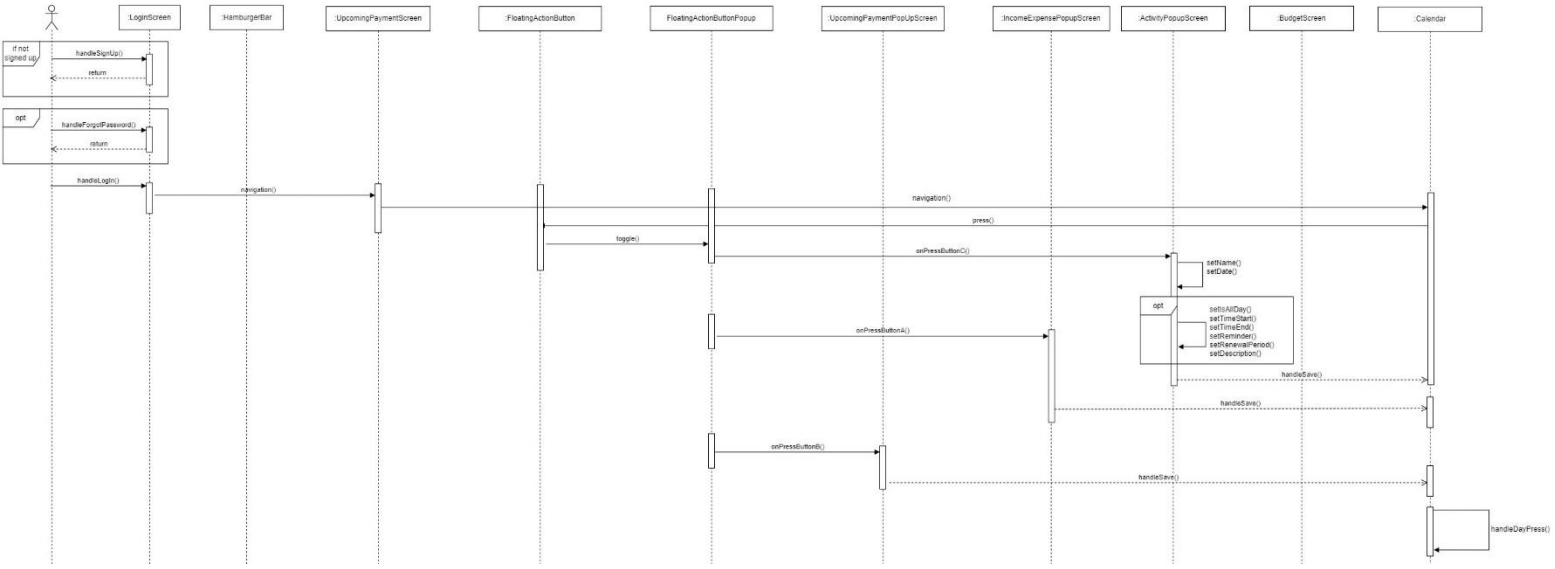


Fig. 4. *Sequence diagram shows the flow of the calendar screen and the activity pop up screen.*

6.2. STATE DIAGRAM

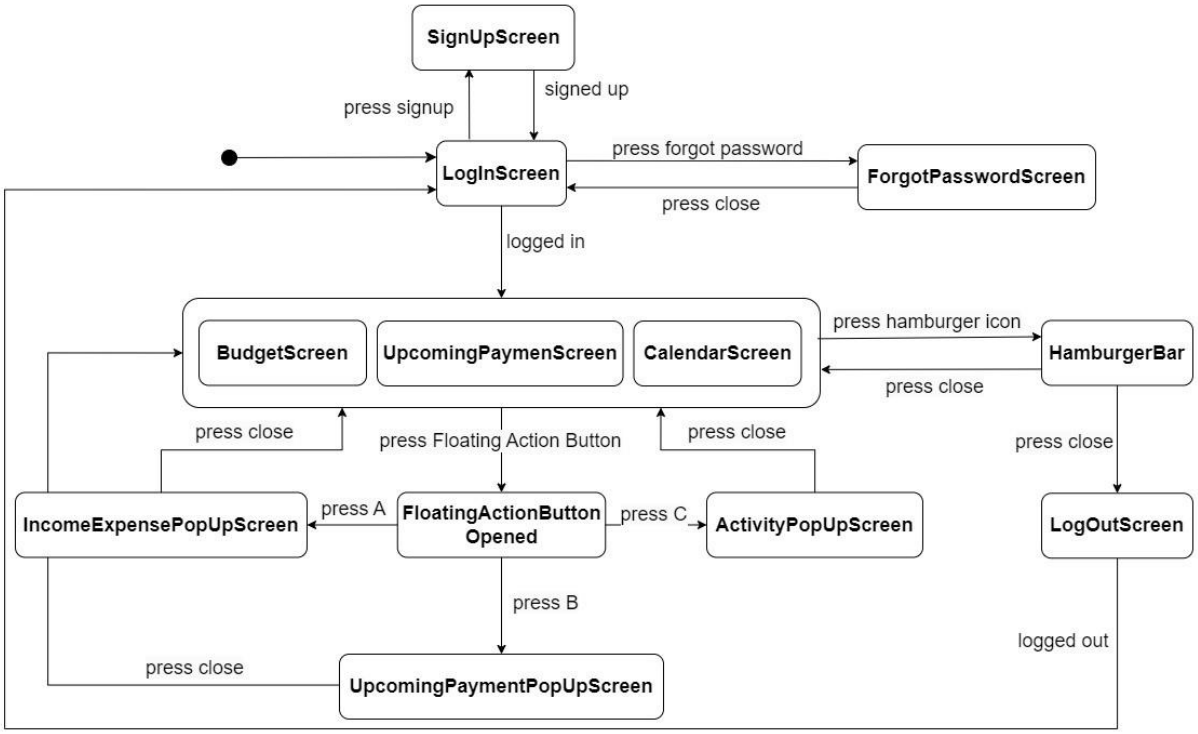


Fig. 5. *State diagram of the system.*

6.3. ACTIVITY DIAGRAMS

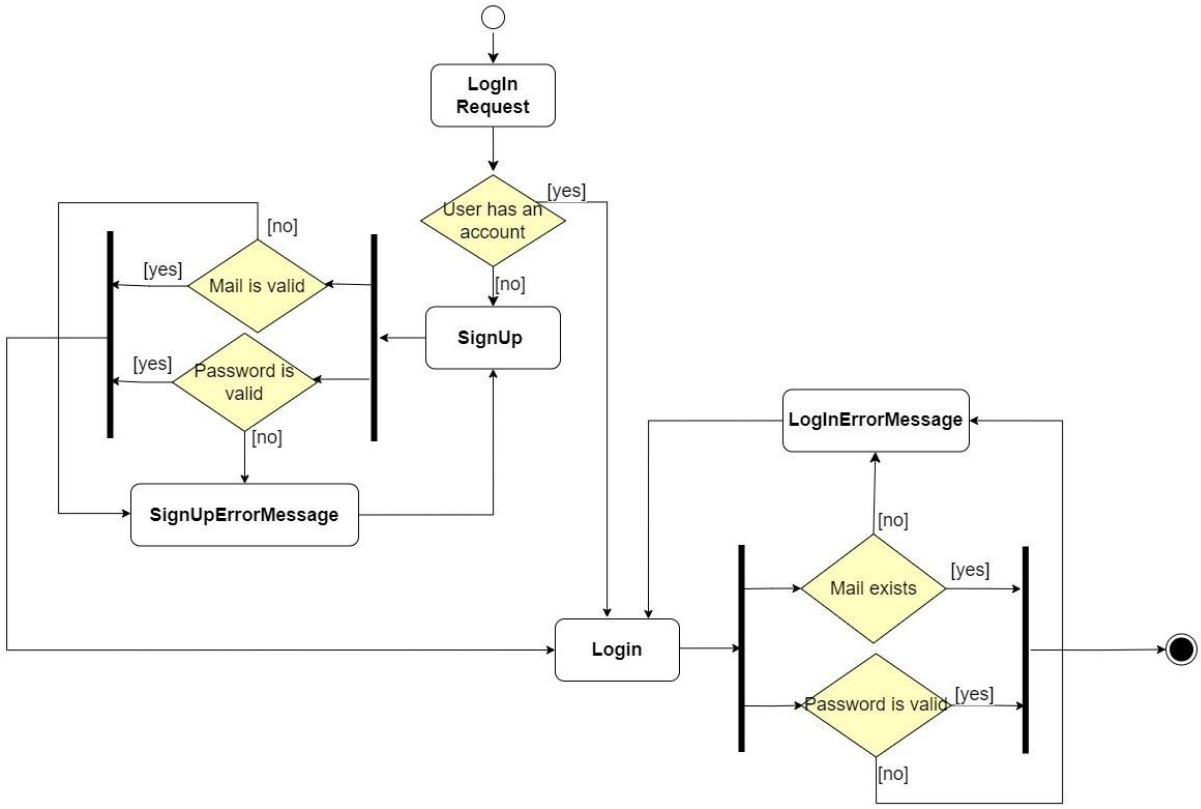


Fig. 6. *Activity diagram of the authentication.*

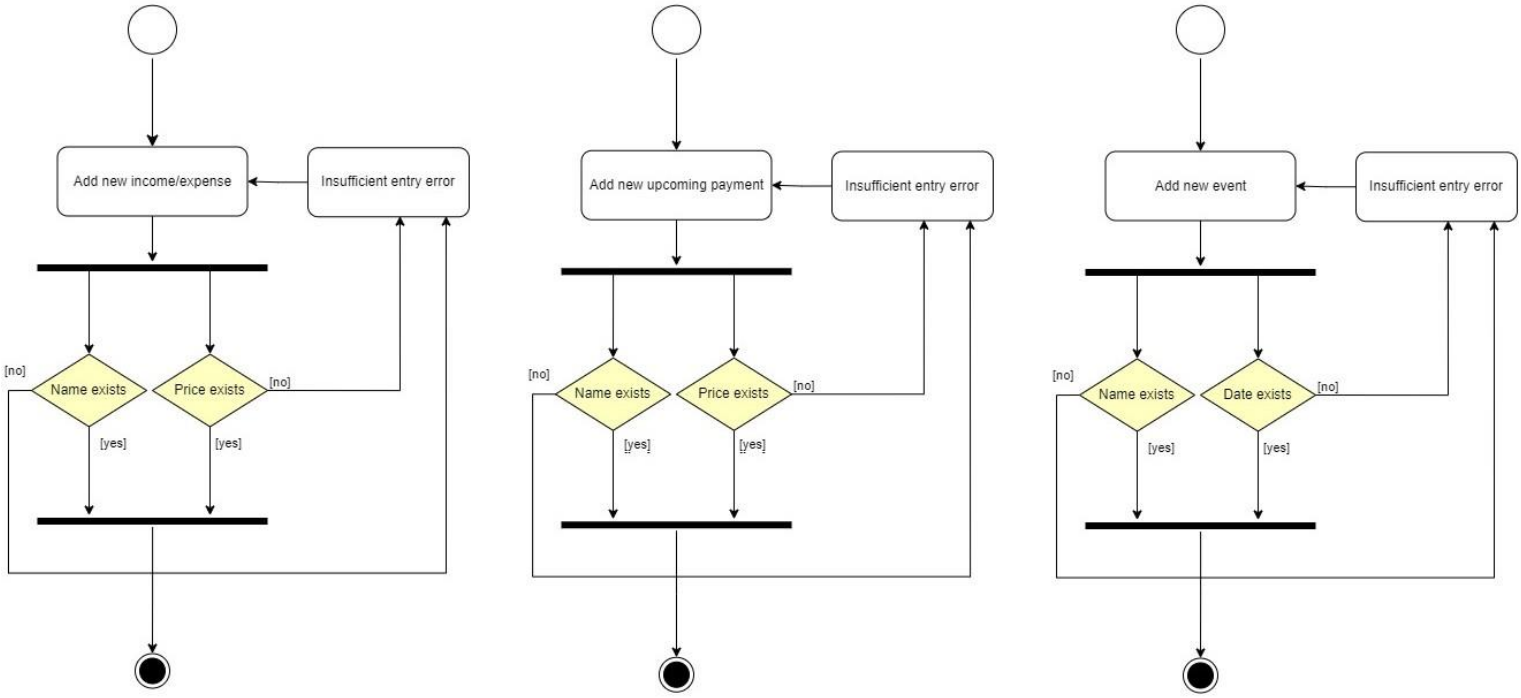


Fig. 7. *Activity diagram of the pop up forms.*

7. SOFTWARE ARCHITECTURE

7.1. UML PACKAGE DIAGRAM

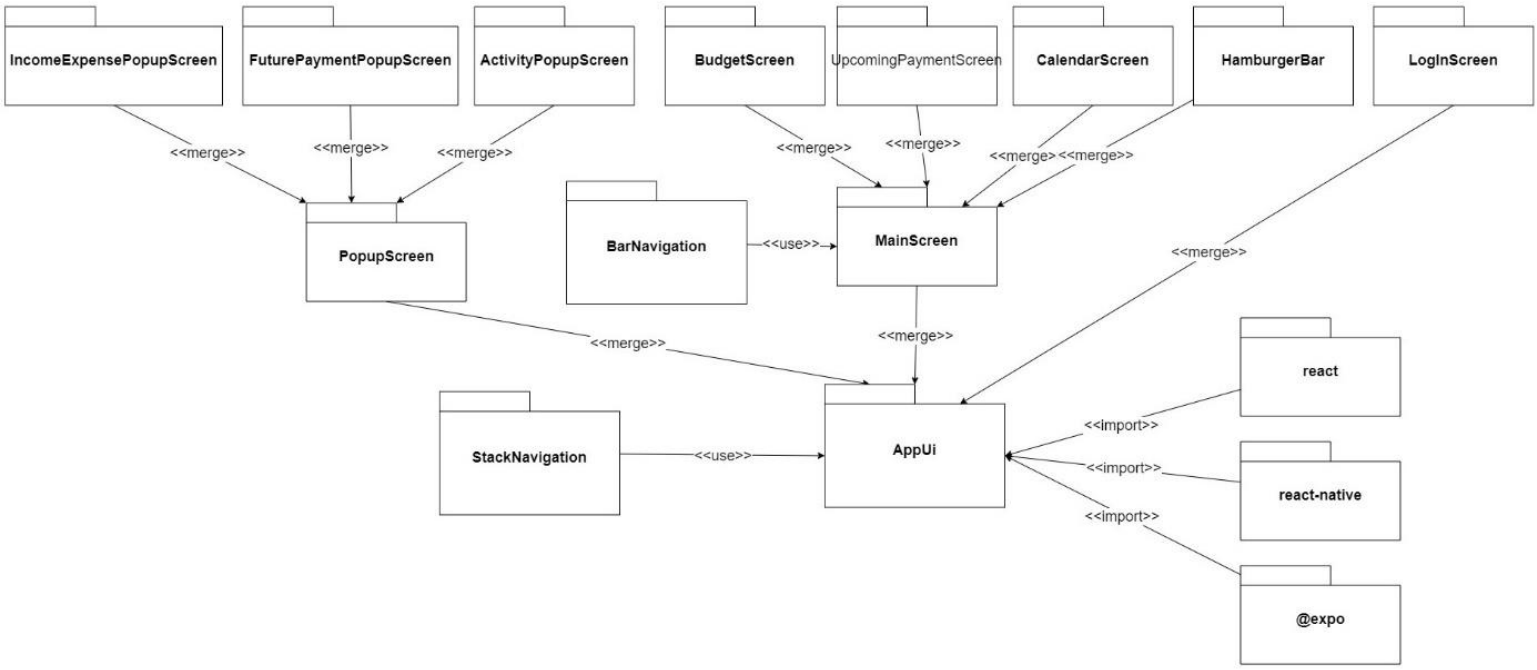


Fig. 8. *UML package diagram of the system.*

7.2. UML COMPONENT DIAGRAM

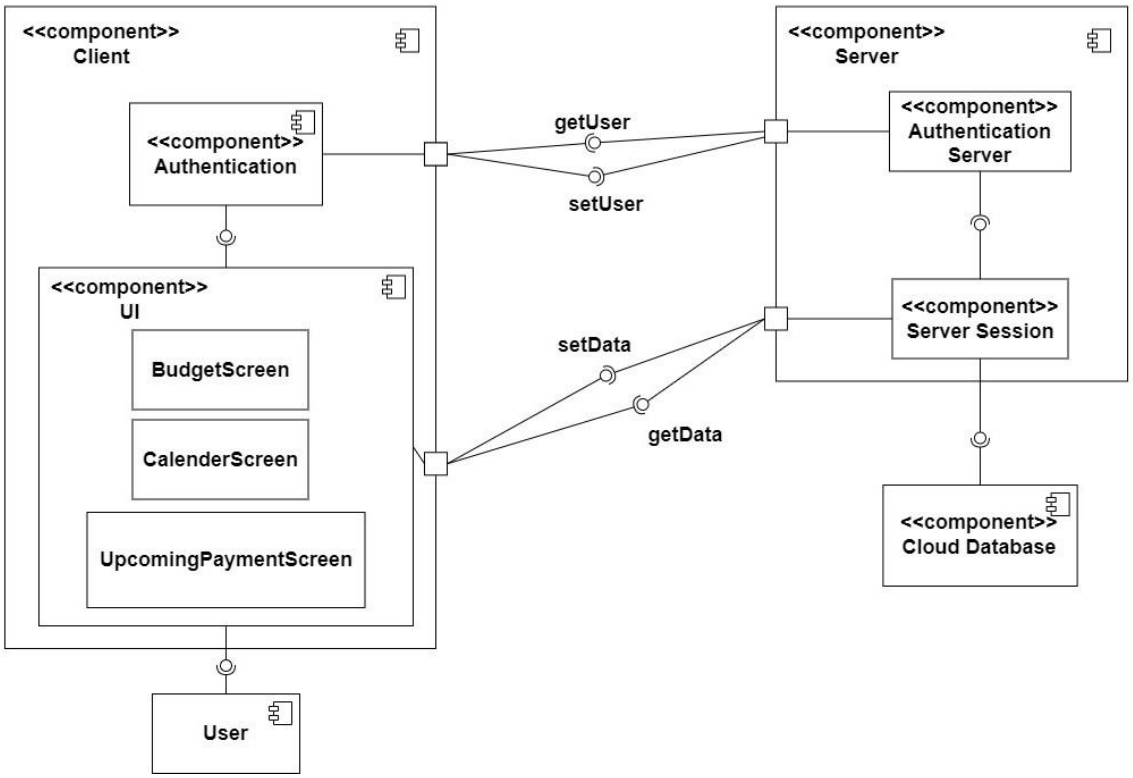


Fig. 9. *Component diagram of the system.*

8. ENTITY RELATIONSHIP (E-R) DIAGRAM

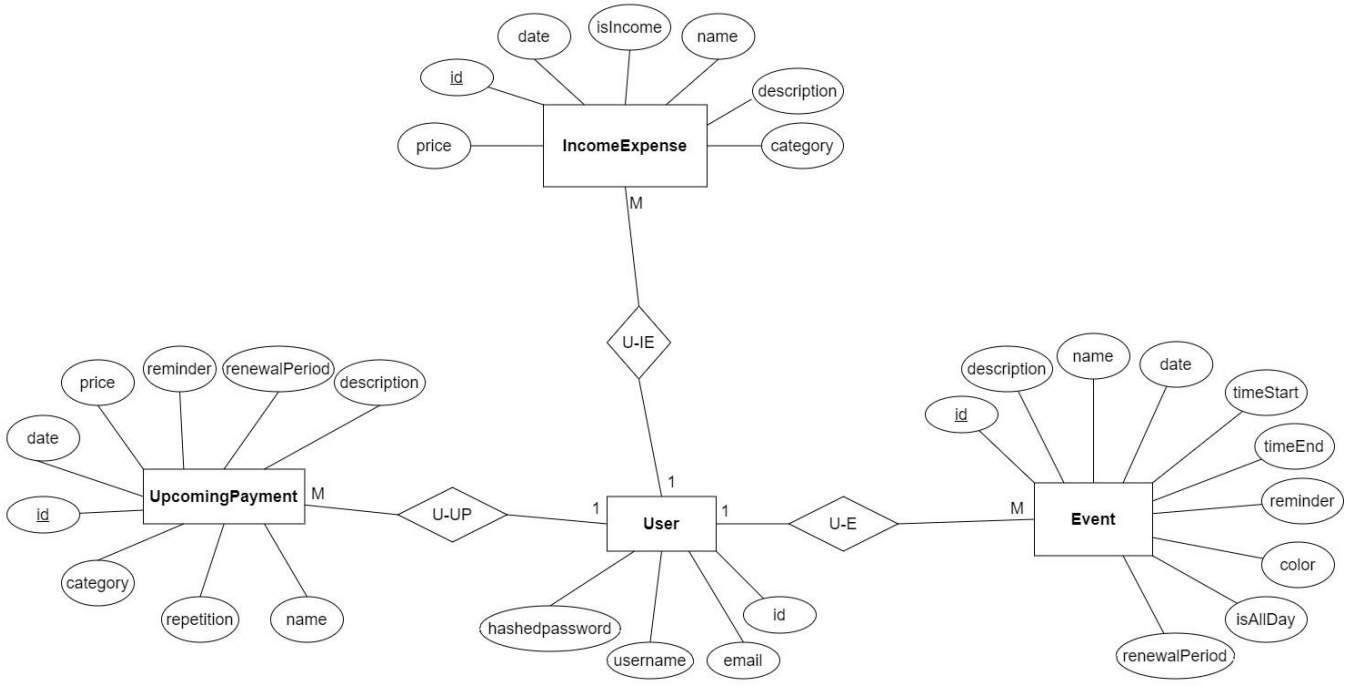


Fig. 10. *E-R diagram of the system.*

9. USER FEEDBACK

The application, satisfies system requirements in a general aspect. Users can sign up via their e-mail addresses, set unique usernames and passwords for security issues. Users can customize the interface based on their preference, but it is limited. However, it can be enhanced. The user can be able to monitor total monthly expense consisting the user's upcoming payments. The user can add payments with various options such as a subscription, an expense or a bill. The user can view the total budget with income and expenses. Also, it is flexible to monitor each one of them monthly. The system differentiates expenses and incomes with different colors. The user can specify both incomes and expenses by giving them specific names and categorize them. The user, be able to monitor the calendar to see their upcoming events. As stated, the system satisfies the requirements explained above.

APPENDIX

A. MEETING MINUTES

Meeting - 1 05.03.2023 (without template)

9.00-9.40 decide the problem, talking about plan, to-do list for next meeting

9.40-10.05 CSE Team - what are the technologies to be used, evaluated what can be solutions of the problem that talke

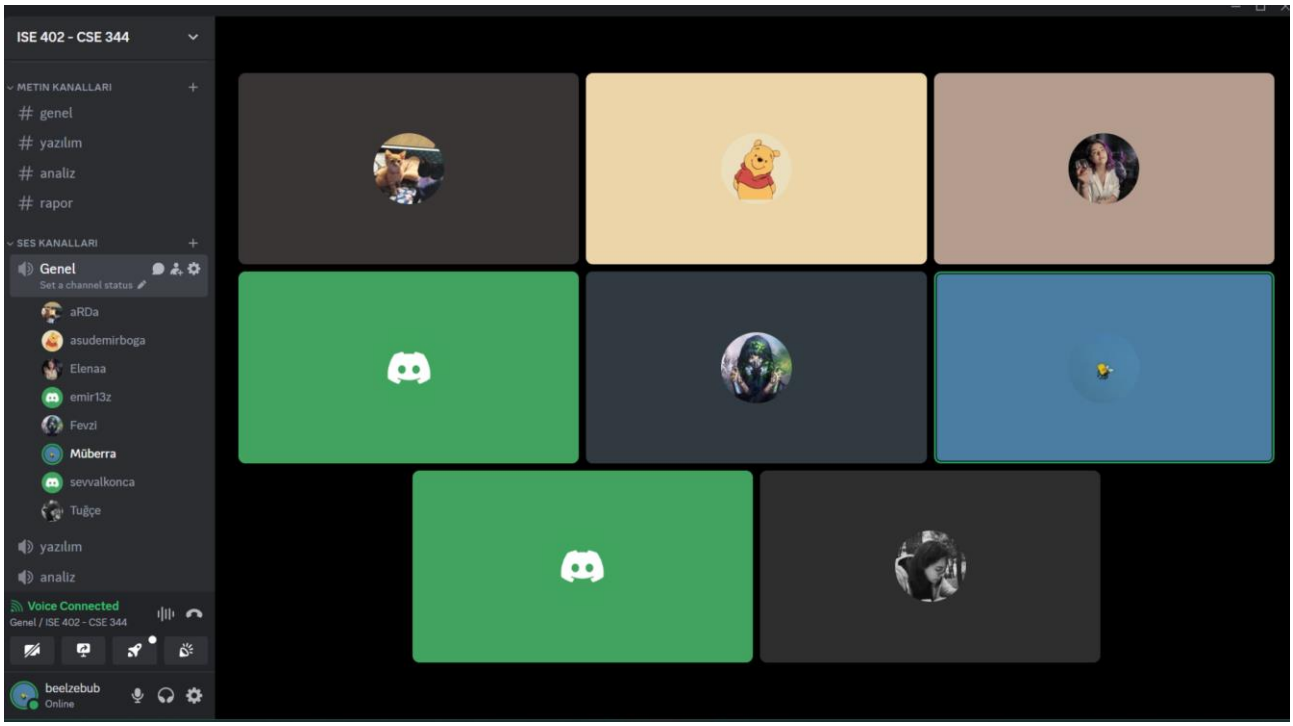


Fig. 11. *Secreenshot of participants for Meeting – 1 .*

Meeting - 2 09.03.2024

Date: 09.03.2024

Time: 21.05 – 22.00

Location: Online Meeting (Google Meet)

Attendees:

Computer Science Engineering Students: Arda Irmak, Asude B. Demirboğa, Fevzi Babaoğlu, Hatice Müberra Gül

Industrial Engineering Students: Berrak Şevval Konca, Emir Zahid Parto, Lale Elena Arzü Karabulut

Agenda: *Discussed and talked about project descriptions according to teachers' feedbacks. We decided on new project topic and examined UI designs for the idea.*

Meeting Notes: Based on the new project groups new members will be contacted.

Individual Contributions: *Each student should share their proposed contribution to the project.*

- Arda Irmak, Asude B. Demirboğa, Fevzi Babaoğlu, Hatice Müberra Gül will focus on new project idea's use cases.
- Berrak Şevval Konca, Emir Zahid Parto, Lale Elena Arzü Karabulut, Tuğçe Tepeciklioğlu will be responsible for requirements and use case.

Next Steps:

- *It should be decided that the team would reconvene on 12.03.2024 at 21.00 to review the progress, address challenges, and plan the next steps.*

Action Items:

- Arda Irmak, Asude B. Demirboğa, Fevzi Babaoğlu, Hatice Müberra Gül to review the project idea with the TA and inform the new group member about the project by 11.03.2024.
- Berrak Şevval Konca, Emir Zahid Parto, Lale Elena Arzü Karabulut, Tuğçe Tepeciklioğlu to communicate with Eylül Damla Gönül Sezer about the new project idea by 11.03.2024.

Submitted by: Berrak Şevval Konca Industrial Engineer Group Member
berraksevval.konca@std.yeditepe.edu.tr

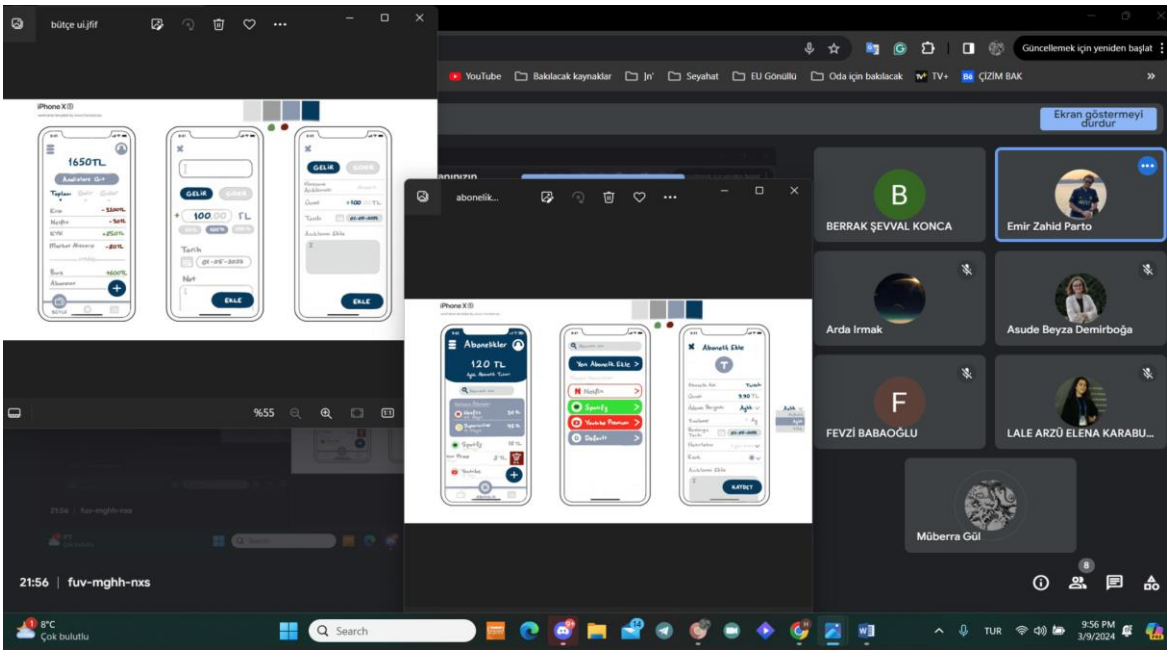


Fig. 12. *Secreenshot of participants for Meeting – 2.*

Meeting Minute – 3 12.03.2024

Date: 12.03.2024

Time: 21.05 – 21.40

Location: Online Meeting (Discord)

Attendees:

Computer Science Engineering Students: Arda Irmak, Asude B. Demirboğa, Fevzi Babaoğlu, Hatice Müberra Gül

Industrial Engineering Students: Berrak Şevval Konca, Emir Zahid Parto, Lale Elena Arzü Karabulut, Tuğçe Tepeciklioğlu

Agenda: *Discussed and talked about use case diagram, next steps, responsibilities of the majors.*

Meeting Notes: Based on the new project groups new member will be contacted since he didn't contacted with project group.

Individual Contributions: *Each student should share their proposed contribution to the project.*

- Arda Irmak, Asude B. Demirboğa, Fevzi Babaoğlu, Hatice Müberra Gül will focus on UI design and implementation.
- Berrak Şevval Konca, Emir Zahid Parto, Lale Elena Arzü Karabulut, Tuğçe Tepeciklioğlu will be responsible for requirements and use case.

Next Steps:

- *It should be decided that the team would reconvene on 16.03.2024 at 17.00 to review the progress, address challenges, and plan the next steps.*

Action Items:

- Arda Irmak, Asude B. Demirboğa, Fevzi Babaoğlu, Hatice Müberra Gül to review the use cases with the Mert Özkaya and try to contact the new group member about the project by 13.03.2024.
- Berrak Şevval Konca, Emir Zahid Parto, Lale Elena Arzü Karabulut, Tuğçe Tepeciklioğlu to communicate with Eylül Damla Gönül Sezer about the use case subsystems by 15.03.2024.

Submitted by: Berrak Şevval Konca Industrial Engineer Group Member
berraksevval.konca@std.yeditepe.edu.tr

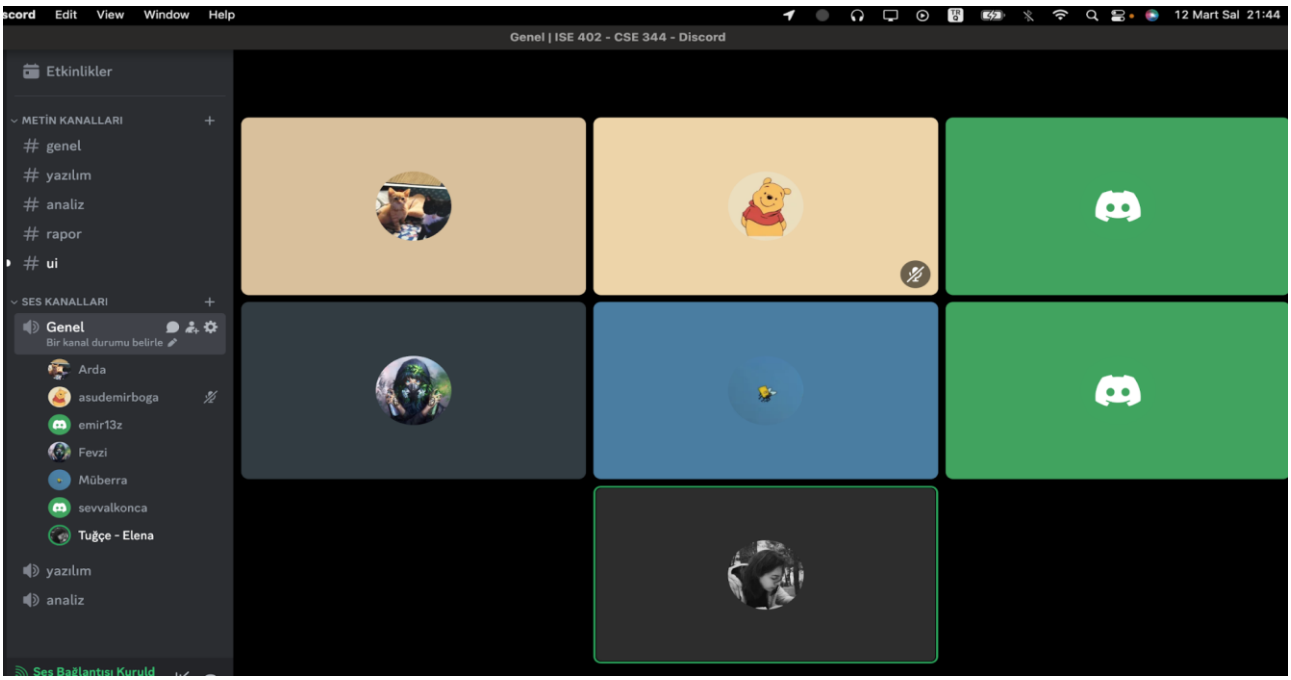


Fig. 13. *Secreenshot of participants for Meeting – 3*

Meeting -4 16.03.2024

Date: 16.03.2024

Time: 16:45 - 17:45

Location: Google Meets

Attendees:

Computer Science Engineering Students: Arda Irmak, Asude B. Demirboğa, Fevzi Babaoğlu, Hatice Müberra Gül

Industrial Engineering Students: Berrak Şevval Konca, Emir Zahid Parto, Lale Elena Arzü Karabulut

Agenda: *Controlled Use Case Diagrams, discussed new features for the idea, talked about next steps. Pivoted the value proposition.*

Meeting Notes: Use Case Diagrams will be completed and Volere for Requirements & Use Case Specification. UI will be changed.

Individual Contributions: *Each student should share their proposed contribution to the project.*

- Arda Irmak, Asude B. Demirboğa, Fevzi Babaoğlu, Hatice Müberra Gül will focus on the UML class diagram. Make the changes on the UI.
- Berrak Şevval Konca, Emir Zahid Parto, Lale Elena Arzü Karabulut, Tuğçe Tepeciklioğlu will be responsible for changing requirements and use case, also will be complete Volere for Requirements & Use Case Specification

Next Steps:

- *There will be last check and discussion via Whatsapp and Discord about diagrams and cases before deadline upload.*

Action Items:

- Arda Irmak, Asude B. Demirboğa, Fevzi Babaoğlu, Hatice Müberra Gül complete on the UML class diagram by 19.03.2024. Make the changes on the UI by 17.03.2024.
- Berrak Şevval Konca, Emir Zahid Parto, Lale Elena Arzü Karabulut, Tuğçe Tepeciklioğlu focus to complete Volere for Requirements & Use Case Specification 19.03.2024.

Submitted by: Emir Zahid Parto Industrial Engineer Group Member
emirzahid.parto@std.yeditepe.edu.tr

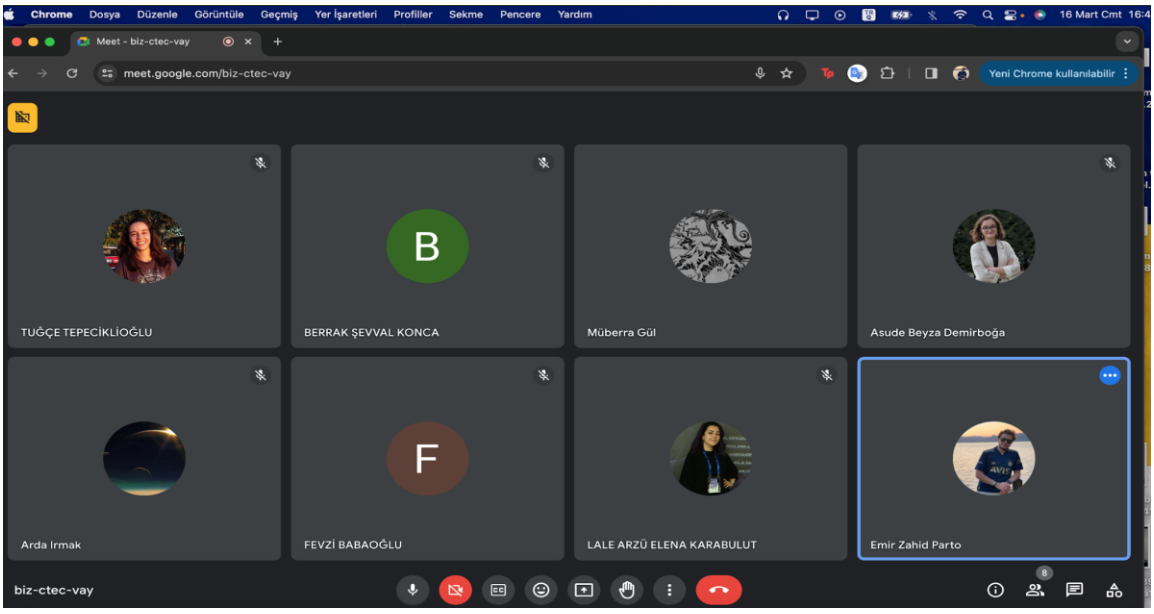


Fig. 14. *Secreenshot of participants for Meeting – 4*

Meeting - 5 27.03.2024

Date: 27.03.2024

Time: 21:10 - 21:40

Location: Discord

Attendees:

Computer Science Engineering Students: Arda Irmak, Asude B. Demirboğa, Fevzi Babaoğlu, Hatice Müberra Gül

Industrial Engineering Students: Tuğçe Tepeciklioğlu, Lale Elena Arzü Karabulut

Agenda: *Shared parts of report that completed, discussed necessary revises, report has been finalized*

Meeting Notes: Checked Volere of Requirements, Use Cases last report, Specification of Requirements

Individual Contributions: *Each student should share their proposed contribution to the project.*

- Arda Irmak, Asude B. Demirboğa, Fevzi Babaoğlu, Hatice Müberra Gül will check finalized report and give feedback if there are.
- Berrak Şevval Konca, Emir Zahid Parto, Lale Elena Arzü Karabulut, Tuğçe Tepeciklioğlu will edit if it need and complete report for final.

Next Steps:

- *Complete report and upload.*

Action Items:

- Arda Irmak, Asude B. Demirboğa, Fevzi Babaoğlu, Hatice Müberra Gül check the documents by 29.03.2024.
- Berrak Şevval Konca, Emir Zahid Parto, Lale Elena Arzü Karabulut, Tuğçe Tepeciklioğlu focus to complete first step report and upload 29.03.2024.

Submitted by: Emir Zahid Parto Industrial Engineer Group Member
emirzahid.parto@std.yeditepe.edu.tr

Meeting - 6 12.05.2024

Date: 12.05.2024

Time: 13:00 - 13:14

Location: Discord

Attendees:

Computer Science Engineering Students: Arda Irmak, Asude B. Demirboğa, Fevzi Babaoğlu, Hatice Müberra Gül

Industrial Engineering Students: Tuğçe Tepeciklioğlu, Berrak Şevval Konca, Emir Zahid Parto

Agenda: *Talked about CSE team's progress on the project and CSE team gave brief to ISE team.*

Meeting Notes: CSE team showed video that they made about interface to ISE team. Discussed project's next steps.

Individual Contributions: *Each student should share their proposed contribution to the project.*

- Arda Irmak, Asude B. Demirboğa, Fevzi Babaoğlu, Hatice Müberra Gül told about project's software details. They will complete software part.
- Berrak Şevval Konca, Emir Zahid Parto, Tuğçe Tepeciklioğlu took brief about project's software part. They will start to write project's final report.

Next Steps:

- *Complete application and final report.*

Action Items:

- Arda Irmak, Asude B. Demirboğa, Fevzi Babaoğlu, Hatice Müberra Gül keep progress the application by 29.05.2024.
- Berrak Şevval Konca, Emir Zahid Parto, Tuğçe Tepeciklioğlu focus to complete final report.

Submitted by: Emir Zahid Parto Industrial Engineer Group Member
emirzahid.parto@std.yeditepe.edu.tr

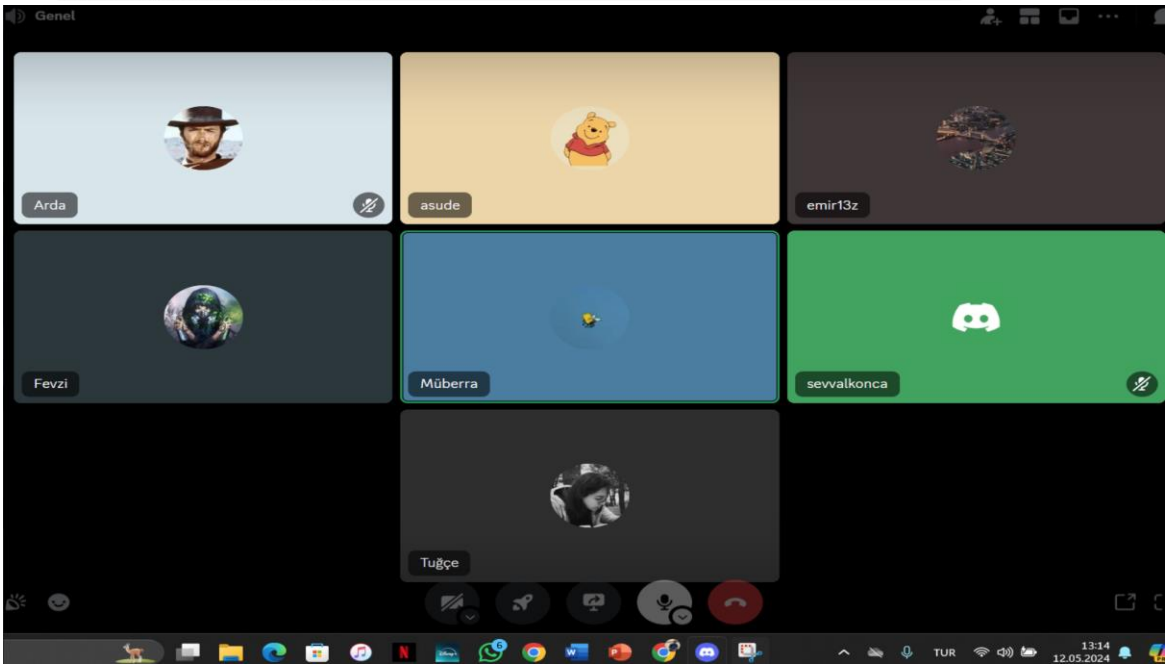


Fig. 15. *Secreenshot of participants for Meeting – 6 .*

Meeting - 7 25.05.2024

Date: 25.05.2024

Time: 18:05 – 18:25 / 19:00-19:30

Location: Discord

Attendees:

Computer Science Engineering Students: Asude B. Demirboğa, Fevzi Babaoğlu, Hatice Müberra Gül

Industrial Engineering Students: Berrak Şevval Konca, Emir Zahid Parto, Lale Elena Arzü Karabulut, Tuğçe Tepecikliođlu

Agenda: *Discussed and talked about final report and poster of the project.*

Meeting Notes: The final report and poster were checked. Revisions and things need to be added were talked. Conclusion and user feedback discussed.

Individual Contributions: *Each student should share their proposed contribution to the project.*

- Arda Irmak, Asude B. Demirboğa, Fevzi Babaoğlu, Hatice Müberra Gül will focus to complete app.
- Berrak Şevval Konca, Emir Zahid Parto, Lale Elena Arzü Karabulut, Tuğçe Tepecikliođlu will be responsible to complete report and poster.

Next Steps:

- *It should be completed of project presentation. Report and poster will be completed for upload.*

Action Items:

- Arda Irmak, Asude B. Demirboğa, Fevzi Babaoğlu, Hatice Müberra Gül will focus to complete duties need to be done.
- Berrak Şevval Konca, Emir Zahid Parto, Lale Elena Arzü Karabulut, Tuğçe Tepecikliođlu will focus to complete report and poster by 28.05.2024.

Submitted by: Emir Zahid Parto Industrial Engineer Group Member
emirzahid.parto@std.yeditepe.edu.tr

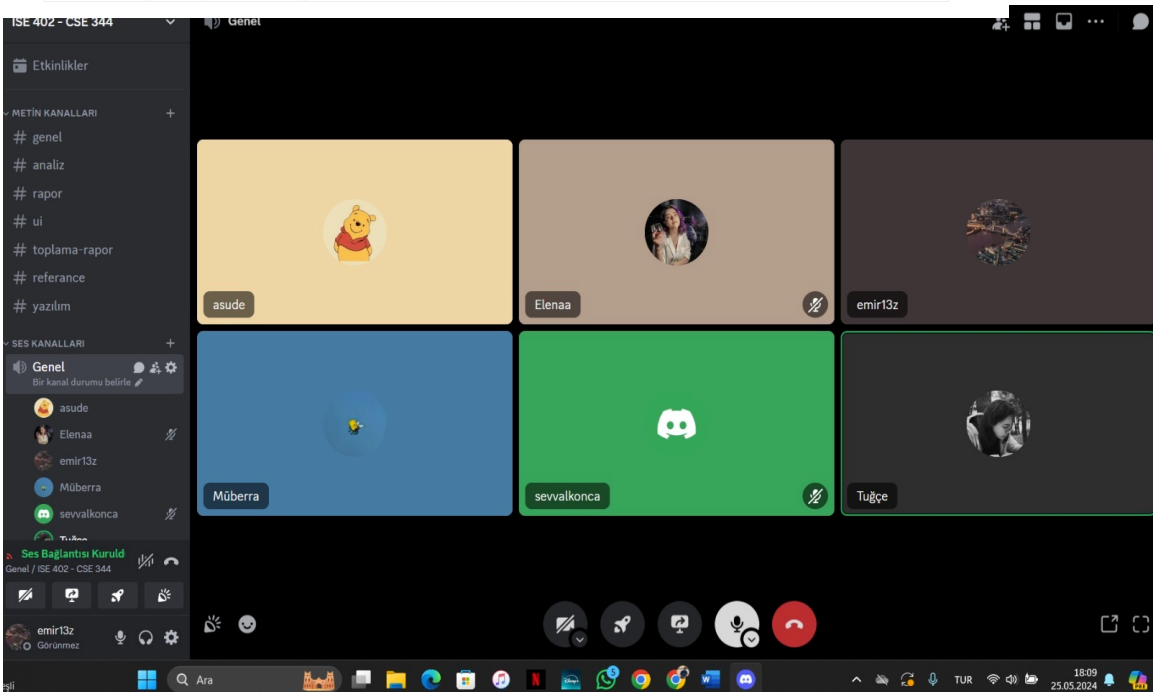


Fig. 16. *Secreenshot of participants for Meeting – 7*

GLOSSARY & REFERENCES

[1] Lusardi, A., & Tufano, P. (2009). "Debt Literacy, Financial Experiences, and Overindebtedness." NBER Working Paper No. 14808.

[2] Pousttchi, K., & Dehnert, M. (2018). "Exploring the digitalization impact on consumer decision-making in retail banking." Electronic Markets, 28(3), 265-286.

[3] Berg, T., Burg, V., Gombović, A., & Puri, M. (2020). "On the Rise of FinTechs: Credit Scoring using Digital Footprints." Review of Financial Studies, 33(7), 2845-2897.

[4] Ozkaya, Mert. (2024). CSE344: Software Engineering [Lecture Notes]. Yeditepe University.

[5] Goularas, Dionysis. (2024). CSE348: Database Management Systems [Lecture Notes]. Yeditepe University.