Software Requirements Specification (SRS) Document

MonkTraderApp™

Prepared by **Abiodun Kumuyi**

Trade Monastery/ALU

06-02-2024

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document outlines the requirements for the **MonkTraderApp™**, a web-based platform designed to provide financial market reports, trading signals, and educational resources. This document covers the initial release (Version 1.0) of the MonkTraderApp™. It includes the functionalities related to user registration, content management by superusers, and content consumption by regular users. The scope of this SRS is limited to these core functionalities and does not include peripheral systems or external integrations that may be part of future expansions.

1.2 Document Conventions

This section describes the standards and typographical conventions followed in writing this SRS document. These conventions are intended to ensure clarity, consistency, and ease of understanding.

Standards

FR: Functional Requirement

NFR: Non-Functional Requirement

UI: User Interface

1. **Text Formatting:**
    - **Headings**: Bold and in a larger font size.
    - **Subheadings**: Bold.
    - **Body Text**: Regular font.
2. **Numbering:**
    - Major sections are numbered 1, 2, 3, etc.
    - Subsections are numbered as 1.1, 1.2, 1.3, etc.
    - Sub-subsections are numbered as 1.1.1, 1.1.2, 1.1.3, etc.
3. **Lists:**
    - **Bulleted lists** are used for items without a specific sequence.
    - **Numbered lists** are used for items with a specific order.

Typographical Conventions

1. **Keywords:**
    - **Shall**: Indicates a mandatory requirement.
    - **Should**: Indicates a recommendation.
    - **May**: Indicates an optional feature.
2. **Emphasis:**
    - **Bold**: Used for section titles and important terms.
    - *Italic*: Used for emphasis or to denote document titles.
    - **Underlined**: Used sparingly for emphasis.
3. **Code and References:**
    - Inline code or commands are presented in monospace font.
    - Document references are cited in the format [Author, Year].

Inheritance of Priorities

- Priorities for higher-level requirements are assumed to be inherited by detailed requirements unless explicitly stated otherwise.
- Each requirement statement is assigned its own priority, with the convention:
    - **High**: Critical for the system's functionality.
    - **Medium**: Important but not critical.
    - **Low**: Nice to have, but not essential.

This document adheres to these conventions to maintain a structured and consistent format, facilitating ease of reading and comprehension for all stakeholders.

1.3 Intended Audience and Reading Suggestions

This document is intended for various readers, each with specific needs and interests. The suggested reading sequence is tailored to address the needs of each type of reader:

1. **Developers:** Focus on Chapters 4 and 5, as these detail the functional and non-functional requirements for system design and implementation.
2. **Project Managers:** Start with Chapter 1 for an overview, then proceed to Chapters 4 and 5 to understand the requirements and constraints.
3. **Marketing Staff:** Read Chapter 1 to understand the purpose, scope, and system overview, which are crucial for crafting marketing strategies.
4. **Users:** The overall description in Chapter 2 and the user interface details in Chapter 3 will provide a comprehensive understanding of the system's features and usability.
5. **Testers:** Focus on Chapters 4 and 5, which outline the functional and non-functional requirements that must be verified.
6. **Documentation Writers:** Begin with Chapter 1 for context, and then review Chapters 4 and 5 to ensure all requirements are accurately documented.

1.4 Product Scope

MonkTraderApp™ is a comprehensive fintech platform aimed at financial enthusiasts, traders, and educators. Its primary purpose is to provide a centralized location for accessing financial market reports, trading signals, and educational materials. The application offers:

- **Superuser Functionality**: Content creation and management capabilities for market reports, trading signals, and educational materials.
- **User Functionality**: Content consumption capabilities for regular users, who can view and interact with the information but cannot modify it.

Relevant benefits include:

- **Enhanced Financial Decision-Making**: By providing timely market reports and trading signals.
- **Educational Value**: Through accessible financial education resources.
- **User Engagement**: A platform that supports continuous learning and informed trading.

1.5 References

The following documents and web addresses are referenced in this SRS:

- **Django Documentation**: https://docs.djangoproject.com/en/stable/
  - *Title*: Django Documentation
  - *Author*: Django Software Foundation
  - *Version*: Stable
  - *Date*: Current as of document creation
- **PEP 8 - Style Guide for Python Code**: https://www.python.org/dev/peps/pep-0008/
  - *Title*: PEP 8 - Style Guide for Python Code
  - *Author*: Python Software Foundation
  - *Version*: N/A
  - *Date*: Current as of document creation
- **MonkTraderApp™ GitHub Repository**: https://github.com/akumuyi/MonkTraderApp.git
  - *Title*: MonkTraderApp™ Repository
  - *Author*: Abiodun Kumuyi
  - *Version*: N/A
  - *Date*: Current as of document creation

2. Overall Description

2.1 Product Perspective

<Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.>

2.2 Product Functions

<Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or object class diagram, is often effective.>

2.3 User Classes and Characteristics

<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy.>

2.4 Operating Environment

<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.>

2.5 Design and Implementation Constraints

<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer's organization will be responsible for maintaining the delivered software).>

2.6 User Documentation

<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>

2.7 Assumptions and Dependencies

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>

3. External Interface Requirements

3.1 User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

3.2 Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

3.3 Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

3.4 Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

4. Requirement Specification

Stakeholder Requirements Specification

Functional Requirements

Req ID	Requirements	Description
FR 1	**User Registration and Authentication**	
FR 1.1	User Registraion	The system shall allow new users to create an account by providing their first name, last name, username, email, and password.
FR 1.2	Secure Login	The system shall allow existing users to log in using their username and password.
FR 2	**Content Management and Access**	
FR 2.1	Superuser Content Management	The system shall allow superusers to create, upload, and manage market reports, trading signals, and educational materials.
FR 2.2	Regular User Content Access	The system shall allow regular users to view market reports, trading signals, and educational materials.
FR 3	**Market Report Generation**	
FR 3.1	Market Report	The system shall store market reports in a database and allow superusers to filter by week, order by descending.
FR 3.2	Trading Signals	The system shall store trading signals in a database and allow superusers to input text area signals.

FR 3.4	Market Relevance	Provide insights on key market trends, potential trade setups, and risk assessments
FR 4	**Educational Resources**	
FR 4.1	Education Materials	The system shall allow superusers to upload educational materials in PDF format, which are stored in a database and embedded in HTML for regular users.
FR 4.2	Demography	Content should be tailored to the specific needs of Africa's low-income earners
FR 4.3	Interactive Elements	Include interactive elements such as trivias and tutorials
FR 5	**Customizable Alerts**	
FR 5.1	Setup Alerts	Allow users to users to setup alerts for specific market changes and report releases
FR 5.2	Notifications	Notifications should be sent via emails, SMS, or in-app alerts
FR 6	**Subscription Management**	
FR 6.1	Subscription Tiers	Implement different subscription tiers (fremium and premium) with varying levels of access to features
FR 6.2	Plan and Payment	Users should be ble to manage the subscription plans and payment details within the platform
FR 7	**Feedback and Support**	
FR 7.1	Feedback Channels	Provide mechanisms for users to submit feedbacks and report issues
FR 7.2	Report Handling	Implement a support system for handling user inquiries and technical support
FR 8	**Administrative Functions**	
FR 8.1	Platform Management	Admins should be able to manage user accounts, subscriptions, and platform content
FR 8.2	Analytics and Reporting	Implement analytics and reporting tools for admins to monitor platform usage and performance

5. Other Nonfunctional Requirements

Requirement Type	Req ID	Description
Security	NFR 1	• Implement strong encryptionfordata storage and transmission • Ensure user data privacy and comply with relevant data protection • Regular audits and vulnerability assessments should be conducted
Performance	NFR 2	• Platform must handle high volume of real-time data without significant delays • Ensure quick load times and responsiveness for user interactions
Scalability	NFR 3	• System should be scalabe to accomodate increasing numbers of users and data volume • Use scalable cloud infrastructure to handle peak loads efficiently
Reliability	NFR 4	• Ensure high availability of the platform with minimal downtime • Implement backup and data recovery plans to safegaurd against data loss
Usability	NFR 5	• User interface should be intuitive and easy to navigate • Provide clear instructions and help resources to assist users • Ensure accessibility for users with disabilities by adhering to relevant accessibility standards
Maintainability	NFR 6	• The codebase should be well documented and follow best practices for maintainability • Implement a continuous integration and deployment pipeline for regular updates and improvements • Provide tools and processes for monitoring and addressing technical issues promptly
Compatibility	NFR 7	• Ensure compatibility with major web browsers and mobile devices • The platform should work seamlessly on different operating systems and screen sizes
Business Rules	• Superuser Permissions	• Only superusers shall be able to create, upload, and manage content on the platform.

Requirement Type	Req ID	Description
	● Regular User Permissions ● Membership Status	● Regular users shall only have permission to view and interact with the content without modification capabilities. ● The system shall differentiate users based on their membership status (premium or freemium) and provide appropriate access levels.

6. Appendix

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

Appendix B: Analysis Models

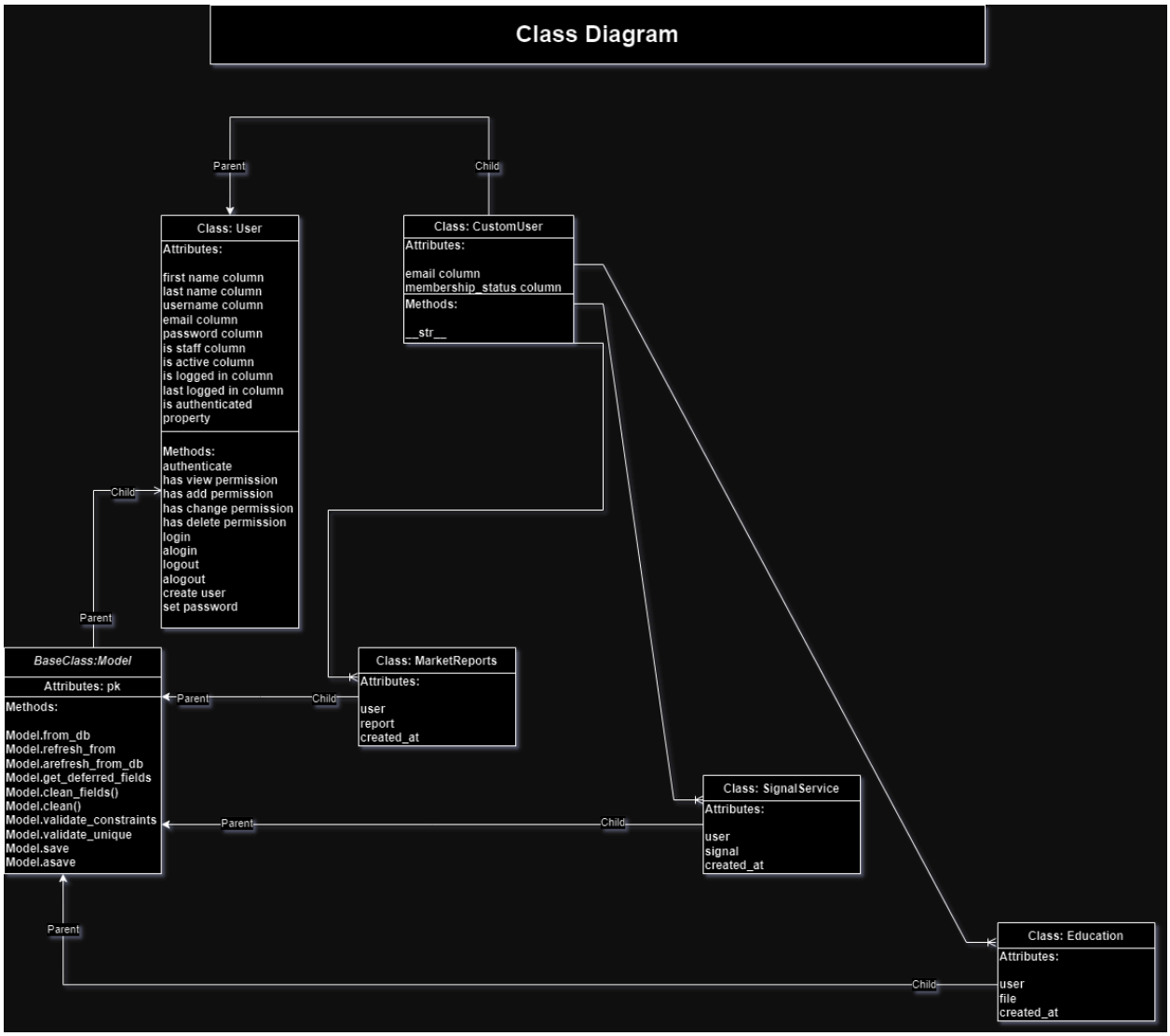


Fig 6.0 Class Diagram

Fig 6.0 above is a class diagram in the Unified Modeling Language (UML), which is a standardized language for software design. It visually shows the relationships between different classes in a system.

In the image, the class diagram shows how the “Custom User” class inherits from the “User” class. The “Custom User” class inherits all the attributes and methods of the “User” class, and it can also have its own specific attributes and methods.

Some of the attributes inherited by the “Custom User” class include first name, last name, username, email, password, is staff, is active, is logged in, and last logged in. It also inherits methods like authenticate, login, alogin, logout, alogout, and create user.

The “Custom User” class also has its own specific attributes, such as membership status, and methods, such as has view permission, has add permission, has change permission, and has delete permission. These methods allow the “Custom User” to perform actions that the “User” class cannot.

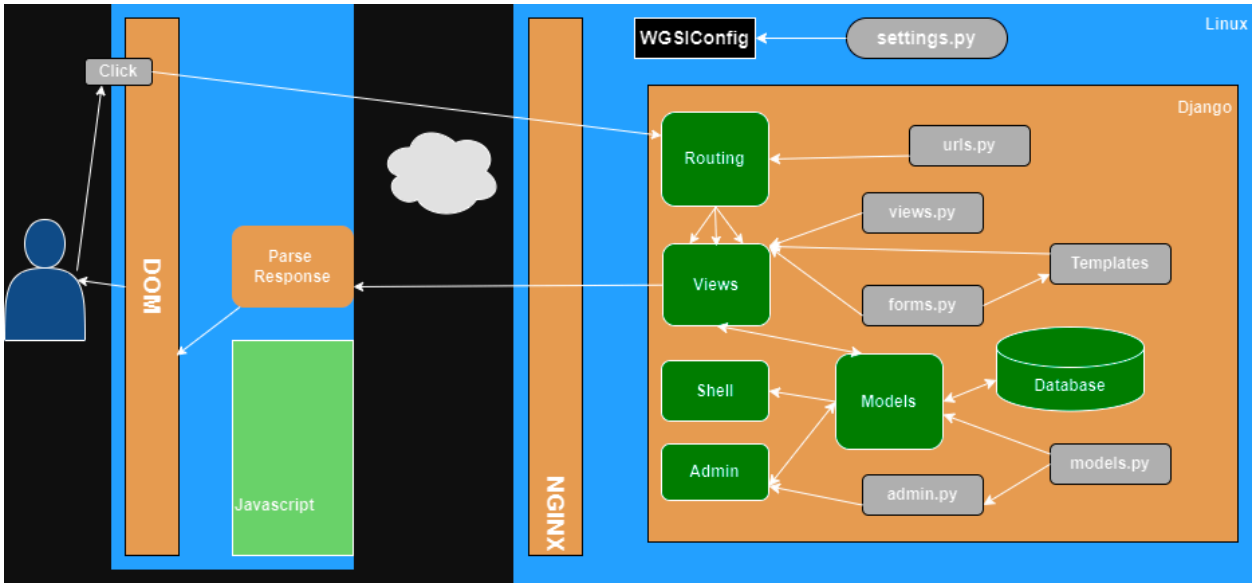


Fig 6.1 Component Diagram

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. A Django web application typically consists of a collection of Python modules. Each module plays a specific role in the overall functionality of the application.

The class diagram above shows some of the classes and their relationships used in a Django web application. Here are some of the classes:

- **WSGIconfig**: This class is the entry point for a WSGI-compatible web application. A WSGI application is a Python application that adheres to the Web Server Gateway Interface (WSGI) standard. WSGI is a specification that describes how a web server should communicate with a web application written in Python.
- **settings.py**: This file contains all the configuration settings of the Django project.
- URLconf (urls.py): This file maps URLs to views functions. A URLconf is a Python module that maps URLs to handler functions, often referred to as views. These views are responsible for processing user requests and returning responses.
- **Views (views.py)**: This file contains the view functions that handle user requests. A view function is a Python function that takes a web request as an argument and returns a web response.
- **Templates**: These are HTML files that define the presentation layer of a Django web application. A Django template is an HTML file that contains special tags that Django can use to insert dynamic content.
- **Models (models.py)**: This file defines the data models of a Django application. A data model is a Python class that represents the structure of your data. A model class essentially defines the fields or attributes of the data and their data types.
- **Admin**: This is the Django administration interface that allows you to manage your data through a user-friendly web interface.
- **Forms (forms.py)**: This file defines forms that are used to collect user input. A Django form is a way to ensure that user input is validated and sanitized before it is used in your application.
- **Database**: This is the backend database that stores the data used by the Django application.
- **NGINX**: This is likely a web server that serves the Django application. A web server is a program that listens for requests from web browsers and delivers web content.

The flow represented in the class diagram is as follows:

1. A user makes a request to a URL.
2. The URLconf (urls.py) maps the URL to a view function in views.py.
3. The view function processes the request and retrieves data from the models.py if needed.
4. The view function may also use a form from forms.py to collect user input.
5. The view function returns a template which may be dynamically rendered using data retrieved from models.py
6. The web server (NGINX) then delivers the response to the user’s web browser.

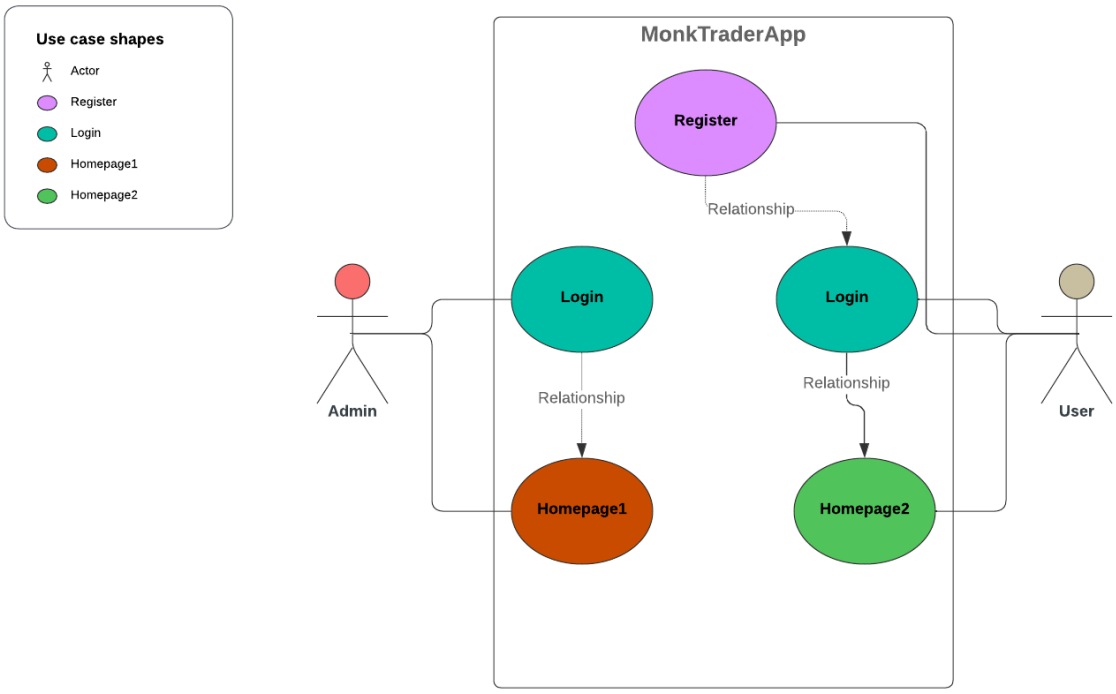


Fig 6.3 Use Case Diagram

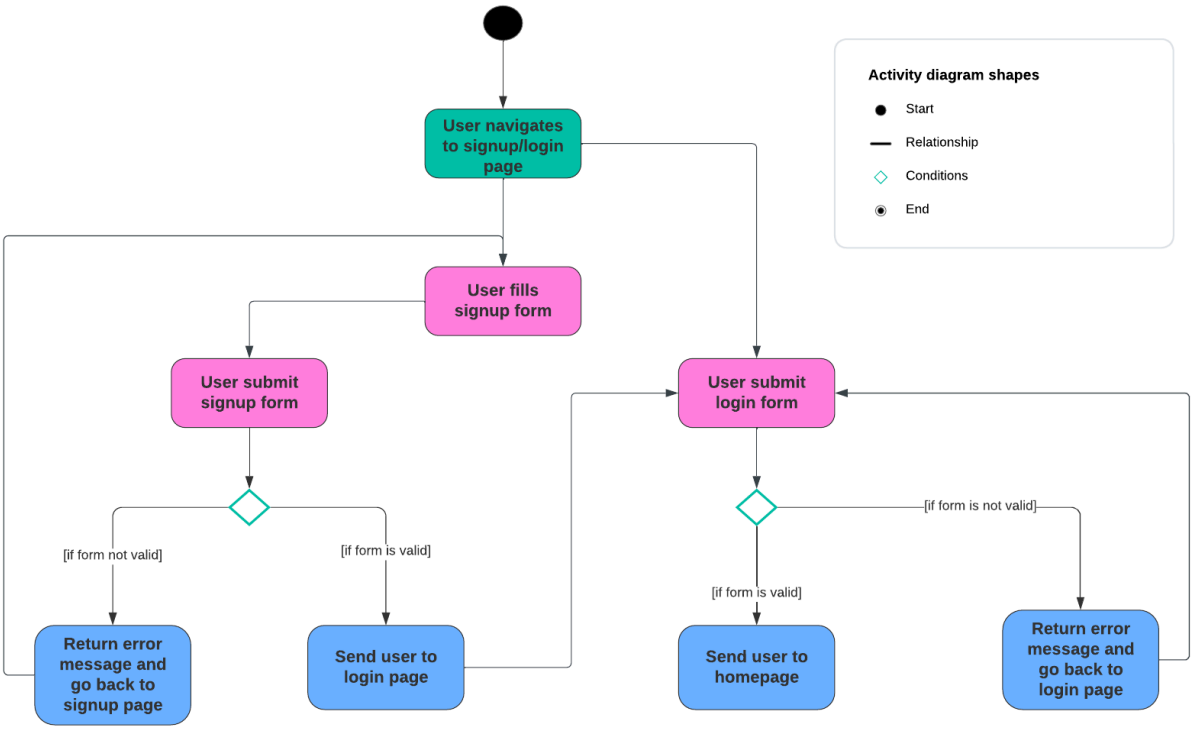
Fig 6.3 is a use case diagram, which is a type of diagram in the Unified Modeling Language (UML). It is used to visualize the interactions between actors (users) and a system to achieve specific goals.

The actors in the diagram are Admin and User. The system is Monk TraderApp.

The use cases in the diagram are Register, Login, and Homepage(1&2).

The relationships between the actors and the system are shown by dotted lines connecting the actors to the use cases. For example, the User actor has a relationship with the Register use case, which means that the User actor can register for the Monk TraderApp system.

The relationships between the use cases are shown by lines connecting the use cases. For example, there is a line connecting the Login use case to the Homepage2 use case, which means that a user can login to the system and then access the Homepage2.



Activity Diagram

Fig 6.4 Activity Diagram (User Registering Example)

Fig 6.4 visually depicts the stepwise activities involved in a process, along with the conditions that influence the flow between those steps. The activity diagram shows the process for a user taking an action and then returning to the login page.

Here's a breakdown of the activities and conditions:

1. **Start:** The process begins.
2. **User navigates to signup/login page:** The user starts by going to the signup/login page.
3. **Condition:** The system checks whether the user is signing up or logging in.
    - **Yes (User fills signup form):** If signing up, the user fills out the signup form.
    - **No (User submits login form):** If logging in, the user submits the login form.
4. **User submits form:** After filling out the form, the user submits it.
5. **Condition:** The system checks whether the submitted form is valid.
    - **No (Return error message and go back to signup page):** If the form is not valid, an error message is shown, and the user is directed back to the signup page.
    - **Yes (if form is valid):** If the form is valid, the process continues.
6. **[if form is valid] Send user to login page (for signup) or homepage (for login):** Depending on signup or login, the system sends them to either the login page or the homepage.
7. **End:** The process ends here.