# **OBJECTIVES**

Determine the purpose and goals of your software.

Identify the specific features and functionalities you want to include.

1. Objective: Develop a content curation software for securities with a focus on providing curated updates and market data to both technical and non-technical users.

Sub-objectives:

* Implement user segmentation into technical and non-technical groups.
* Address unique requirements for each user group while providing shared functionality.

1. Objective: Provide users with the ability to define the frequency of updates.

Sub-objectives:

* Allow users to choose between daily and weekly updates.
* Implement mechanisms to deliver updates according to user preferences.

1. Objective: Enable users to retrieve the latest visual snapshot of market data for any stock if available via the API and website.

Sub-objectives:

* Integrate with the appropriate API(s) to fetch market data.
* Develop a user interface to display visual snapshots of market data.

1. Objective: Facilitate web service and email service for users, requiring technical users to obtain an API key.

Sub-objectives:

* Implement authentication and authorization mechanisms for technical users to obtain and use API keys.
* Enable technical users to access the web service and receive updates via email.

1. Objective: Allow users to unsubscribe from the service at any time.

Sub-objectives:

* Provide a user-friendly option for users to unsubscribe from updates.
* Implement the necessary backend functionality to handle user unsubscribing.

1. Objective: Enable users to define their own portfolio of stocks for personalized updates.

Sub-objectives:

* Develop a feature that allows users to select and manage their subscribed stocks.
* Customize the content updates based on the user's portfolio selections.

1. Objective: Include credible news in the daily and weekly digests, with the ability to exclude daily news from the weekly digest.

Sub-objectives:

* Integrate a news feed or news API to retrieve relevant news articles.
* Implement logic to include news in the digests based on user preferences and frequency settings.

1. Objective: Include the most time-consuming articles in the weekly digest.

Sub-objectives:

* Implement a mechanism to identify time-consuming articles based on criteria such as length or complexity.
* Include these articles in the weekly digest for users who prefer more detailed content.

1. Objective: Capture portions of original articles and provide direct links to the source.

Sub-objectives:

* Implement a mechanism to extract snippets from the original articles.
* Generate links to the source articles for users to access the full content.

1. Objective: Optionally provide the digest via an API.

Sub-objectives:

* Develop an API endpoint to deliver the digest content.
* Implement necessary authentication and authorization mechanisms for API access.

1. Objective: Include cryptocurrency market data and related digests.

Sub-objectives:

* Integrate with appropriate cryptocurrency market data sources.
* Provide curated updates and digests for cryptocurrency information.

1. Objective: Offer the popular portfolio, which captures the most popular stocks per day.

Sub-objectives:

* Implement a mechanism to identify and track popular stocks.
* Provide a curated portfolio of popular stocks for users to view.

1. Objective: Allow non-registered users to view the latest available visual snapshot and daily digest of the popular portfolio on a public website.

Sub-objectives:

* Develop a public website interface to showcase the latest visual snapshot and daily digest for the popular portfolio.
* Ensure access to this information without requiring user registration.

# **ARCHITECTURE**

Design the system architecture, including the backend, frontend,

and any other components. Decide on the technologies and frameworks you want to use (e.g.,

programming languages, databases, web frameworks).

1. Client Interface: Determine the platforms through which users will access your software. This can include web interfaces, mobile applications, or both. Decide on the technologies and frameworks you will use to develop the client interface.
   1. Desktop
   2. Web
2. Backend: Choose a backend framework or technology stack that suits your requirements. This may include server-side programming languages (e.g., Python, Node.js, Java), frameworks (e.g., Flask, Django, Express.js), and databases (e.g., MySQL, PostgreSQL, MongoDB) for storing user data, content metadata, and configurations.
   1. Python
   2. Django
   3. MySQL and PostgreSQL
3. API Integration: Identify the APIs you will integrate with to fetch market data, news articles, and other relevant content. Research and choose reliable and reputable APIs that provide the required data and have good documentation and support.
   1. To check as yet!!!
4. Authentication and Authorization: Implement user authentication and authorization mechanisms to ensure secure access to the system. Determine whether you will use a third-party authentication service (e.g., OAuth) or build your own authentication system.
   1. Django-admin authentication system
5. Database Design: Design the database schema to store user information, user preferences, content metadata, and any other relevant data. Consider the relationships between different entities and the required data integrity.
6. Content Curation Logic: Define the algorithms and rules for curating content based on user preferences, frequency settings, portfolio selections, and popularity. Determine how you will fetch, filter, categorize, and present the content to users.
   1. Plan algorithm
7. Third-Party Services: Identify any additional third-party services or tools you may need for functionalities like email delivery, analytics, or content management. Research and select reliable services that align with your requirements.
   1. To research
8. Scalability and Performance: Consider the potential growth and scalability of your software. Plan for handling increasing user loads, concurrent requests, and data storage requirements. Explore techniques like caching, load balancing, and database optimization to ensure performance.
9. System Integration: Determine if there are any existing systems or platforms you need to integrate with, such as CRM systems, payment gateways, or social media platforms. Assess the necessary integration points and plan the implementation accordingly.
10. Development Methodology: Decide on the development methodology you will follow, such as Agile, Scrum, or Waterfall. Define the development phases, milestones, and iterations to track progress and ensure timely completion.
11. Deployment and Infrastructure: Choose a hosting environment for deploying your software. Consider whether you will use cloud platforms (e.g., AWS, Google Cloud, Azure) or self-hosted servers. Determine the infrastructure requirements and plan the deployment process.
12. Testing and Quality Assurance: Plan for thorough testing of your software. Define test cases, conduct unit tests, integration tests, and end-to-end tests. Consider implementing continuous integration and automated testing to ensure code quality.
13. Documentation and Maintenance: Create documentation for developers, administrators, and end-users. Document the system architecture, API usage, setup instructions, and troubleshooting guidelines. Plan for ongoing maintenance, bug fixes, and future enhancements.
14. Remember, the architectural plan will serve as a blueprint for your development process. It's important to adapt and refine the architecture as you progress and gain more insights. Collaborate with your team members to ensure alignment and make informed decisions.

# **COLLECT AND ORGANIZE CONTENT**

Determine the sources from which you want to curate content.

Implement mechanisms to fetch, extract, and store relevant content from these sources.

Develop algorithms or rules for content categorization, tagging, or ranking.

# **USER MANAGEMENT AND AUTHENTICATION**

Implement user registration, login, and authentication mechanisms.

Define user roles and permissions to control access to different features.

# **USER INTERFACE**

Create an intuitive and user-friendly interface for users to interact with the software.

Design screens for content browsing, searching, and organizing. Implement features like bookmarking, saving,

and favouriting content.

# **CONTENT RECOMMENDATION**

Implement algorithms or mechanisms to recommend relevant content to users

based on their preferences, history, or user-defined criteria. This can involve techniques like

collaborative filtering, content-based filtering, or machine learning algorithms.

# **CUSTOMIZATION AND PERSONALIZATION**

Provide options for users to customize their content preferences,

filters, or notifications. Allow users to create personalized collections or playlists of curated content.

# **ANALYTICS AND REPORTING**

Incorporate analytics and reporting features to track user engagement,

content popularity, or other relevant metrics. Generate insights and reports based on the collected data.

# **TESTING AND DEBUGGING**

Perform thorough testing to ensure the software functions as expected.

Identify and fix any bugs or issues that arise during testing.

# **DEPLOYMENT AND MAINTENANCE**

Deploy the software on your preferred hosting environment.

Set up regular maintenance tasks, including monitoring, updates, and backups