VeriTag

Martin Maxim, Nick Bilotti, Jud Turner, Peter Halvorsen

MVP

For the Minimum Viable Product (MVP) of VeriTag, we will focus on completing Epics 1, 3, and 4 before the Rhodes Symposium, ensuring a functional platform with core features. This MVP will not include actual content scraped by the screen-scraper and will instead provide hard coded example articles just to see the website itself. The website will have a simple, intuitive homepage with a browsing section displaying manually loaded article headlines for navigation. Users will be able to read articles in a clean, distraction-free format and engage in discussions through a comment section. To have general user accounts to post comments and vote on others' contributions, while verified users can provide verified comments with rating explanations. To enhance engagement, users will be able to like/dislike verified comments and upvote/downvote general comments, with low-rated ones being automatically removed. This MVP will provide the foundational user experience, interaction, and feedback mechanisms needed for VeriTag to be showcased at the symposium.

Epic 1: User Interface and Navigation

As a user, I want a well-structured and intuitive interface so that I can easily navigate the website and access content without confusion.

User Stories:

1. As a user, I want to open the website and access a simple and intuitive homepage so that I can quickly understand the layout and find what I need.
2. As a user, I want a browsing section that lists headlines from multiple sources so that I can easily explore different articles.
3. As a user, I want to access and read entire articles in a clean, readable format so that I can focus on the content without distractions.
4. As a user, I want to see general comments and engage with discussions on articles so that I can participate in conversations and understand diverse perspectives.
5. As a programmer of VeriTag, I want to have manually loaded articles as a placeholder for the screen scraper so that I can best design the data in the UI.

Epic 2: Article Data Aggregation and Presentation

As a user, I want regularly updated article data so that I can explore fresh and diverse content with quality ratings.

User Stories:

1. As a user, I want to see at least 5 articles from each of 3–4 sources, updated every 4 hours so that I have access to fresh and diverse content regularly.
2. As a user, I want to see the average total rating for each article on the Browse Articles page so that I can quickly gauge its overall quality.
3. As a user, I want to view detailed category ratings (bias, accuracy, quality) by clicking a pop-up button on the average rating icon so that I can better understand how the article was evaluated.

Epic 3: User Authentication and Interaction

As a user, I want secure authentication and interaction features so that I can engage with content, comment, and contribute reliably

User Stories:

1. As a verified user, I want to log in securely and see my status reflected on the website so that I can access additional features and trust my data is safe.
2. As a general user, I want the ability to post general comments on articles and vote on other users' comments so that I can participate in discussions and share my opinions.
3. As a verified user, I want an additional button to add my own verified comments on articles with ratings for each category and a brief explanation so that I can provide credible and constructive feedback.

Epic 4: User Feedback and Engagement

As a user, I want meaningful feedback and engagement options so that I can influence discussions and improve the platform's content quality.

User Stories:

1. As a user, I want to like or dislike verified user comments so that I can signal their usefulness to other readers.
2. As a user, I want to upvote or downvote general user comments, with those receiving -5 votes automatically deleted so that the quality of discussions is maintained.
3. As a verified user, I want my comments to stand out as trustworthy and informative to other readers so that I can build credibility and add value to discussions.
4. As a user, I want to ensure that article feedback (comments, ratings) contributes to building a helpful reading experience so that the platform continuously improves and meets users' needs.

System Description:

This project is a web platform designed to aggregate articles from multiple sources, encourage meaningful discussions, and provide ratings to evaluate article quality and trustworthiness. It aims to create a better reading experience by integrating user feedback and verified insights.
There are two basic types of users. "Verified" and regular users. Verified users have special access to comment on articles trustworthiness, fact checking, and give ratings based on their expertise. These people are vetted before they get access, and have special access based on their qualifications.

Regular users login and can see the verified users comments on articles, so they can decide what to take out of said articles. If a verified user has something to say about the trustworthiness, facts, or opinions about the article, this can be taken into account from the general user perspective.

Regular users will not be able to edit in the verified comment section or provide ratings. They will have a general comment section below the article, but this is not as serious or qualified as the verified user's contributions.

Functional Requirements:

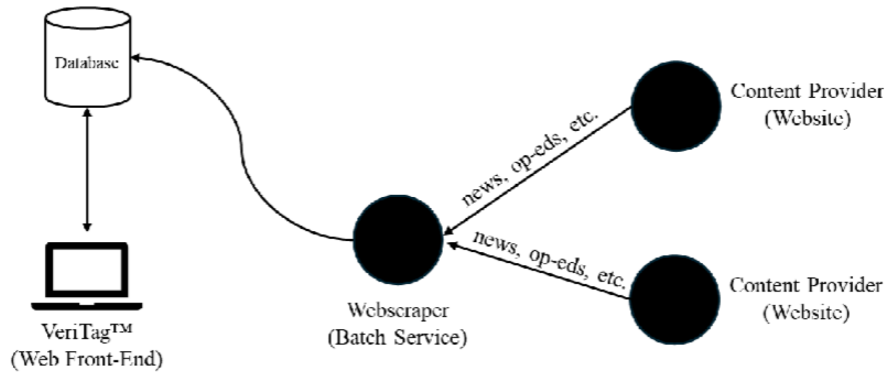
- Open the website
- Browsing section with headlines of articles screen scraped from 3 or 4 different sources
- Get at least 5 articles from every source, update every 4 hours
- On the Browse Articles page, show average total rating for each article
- Pop up button on average rating icon showing averages for each category (bias, accuracy, quality)
- Verified user login in the top right
- Open an article
- See verified user comments on the article with their ratings for each category and a brief explanation
- Can like or dislike the verified user comment
- See whole article
- General comments below
- Upvote / Downvote general user comments (-5 votes deleted)
- If you login as a verified user, basically the same but there is an extra button to add verified users comment

Non Functional Requirements:

- Non-Functional Req. 1 — System Response Time
  The system should respond to user interactions, such as navigating between pages or posting comments, within 2 seconds.

- Non-Functional Req. 2 — Data Freshness
  The system must update article data from all sources every 4 hours to ensure users see the latest headlines and ratings.
- Non-Functional Req. 3 — Availability
  The service must maintain an uptime of at least 99.5% over a 30-day period.
- Non-Functional Req. 4 — Security
  The service must support secure verified user logins by enforcing encrypted connections using HTTPS.
- Non-Functional Req. 5 — Scalability
  The system must handle up to 50 concurrent users without degradation in performance.
- Non-Functional Req. 6 — Legality
  The system must scrape legal content and not infringe on rights of other companies
- Non-Functional Req. 7 — Cordiality
  The system must encourage civil discourse and discourage chaotic and inappropriate dialogue

High-Level System Diagram:


Database
Content Provider (Website)
news, op-eds, etc.
news, op-eds, etc.
Content Provider (Website)
VeriTag™ (Web Front-End)
Webscraper (Batch Service)

Architecture

There are a few basic parts to the architecture.

The web scraper pulls content from the content providers - large media websites. The content gets pushed to the database for Veritag with all the data needed (article name, source, publisher, author, date, summary, and the article itself). It is a screen scraper algorithm that will update in a reasonable time frame to keep the articles current. It's going to pull popular articles and a wide range of topics for people to explore. It pulls this into the Veritag database to be used there.

From there, the web front end of Veritag is the actual user interface. It uses that data from the database pulled by the screen scraper and presents it on the website. So the articles are not from Veritag. But all the operations and users that are using the data are.

Technologies Used

Frontend Development

- **Languages**: HTML, CSS, JavaScript
- **Frameworks/Libraries**: Flask (for backend integration), Bootstrap (for styling and responsiveness)
- **Development Environment**: PyCharm
- **Functionality**: The web front end will provide an interactive user interface for displaying scraped content, search functionality, and potential user interactions.

Backend & Database

- **Language**: Python
- **Framework**: Flask (to handle web requests and serve content)
- **Database**: SQLite (relational database for storing scraped content and metadata)
- **ORM**: SQLAlchemy (for database interactions within Flask)
- **Development Environment**: PyCharm
- **Functionality**: The backend will handle API calls, store content in the database, and manage data retrieval.

Web Scraper

- **Language**: Python
- **Libraries**: …?
- **Development Environment**: PyCharm
- **Functionality**: The web scraper will extract articles from free reputable websites, format the data, and store it in the database for use by the frontend.

Content Providers

- **Initial Sources**: Free articles from reputable websites.
- **Future Plans**: Transition to larger sites with paid content access if necessary.
- **Data Handling**: Scraped content will be processed, filtered, and stored in SQLite for easy retrieval.

Deployment & Hosting

- **Local Development**: PyCharm for coding and testing
- **Potential Hosting**: Flask app deployed on Heroku, AWS, or PythonAnywhere
- **Version Control**: GitHub for source code management

Roadmap 2.0

Sprint 0 (2/6) – Planning & Preparation

- Finalize the planning document, including objectives, scope, and success criteria.
- Define team roles and responsibilities.
- Set up the project repository, development environment, and tracking tools.
- Identify risks and create mitigation strategies.
- Outline detailed requirements for all Epics.

Sprint 1 (2/20) – Initial Development (Half of Epic 1)

- Implement core features of Epic 1 (define key functionalities).
- Set up the database and backend structure if applicable.
- Develop wireframes or basic UI components.
- Establish initial unit tests.
- Conduct code reviews and early testing.

Sprint 2 (3/6) – Complete Epic 1 & Start Epic 3

- Finalize all features of Epic 1.
- Begin implementation of Epic 3 (focus on core logic and backend integrations).
- Refine UI components based on feedback.
- Conduct integration testing for completed features.
- Update documentation as necessary.

Sprint 3 (3/20) – Complete Epic 3

- Finish all functionalities of Epic 3.
- Perform in-depth testing and bug fixes.
- Gather early user feedback if possible.
- Improve performance and efficiency of the system.
- Ensure all Sprint 3 deliverables meet project requirements.

Sprint 4 (4/3) – Begin Epic 4

- Start development of Epic 4, focusing on major components.
- Conduct usability testing and refine UI/UX.
- Address any technical debt from earlier sprints.
- Continue documentation updates.
- Begin working on deployment strategy if applicable.

Sprint 5 (4/15) – Complete Epic 4 & Achieve Minimum Viable Product (MVP)

- Finalize all Epic 4 features.
- Conduct a full round of system testing (functional, integration, performance).

- Ensure MVP is stable and meets project requirements.
- If time permits, begin development on Epic 2 for additional features.

Sprint 6 (4/29) – Final Refinements & Additional Features (Epic 2)

- Finish all remaining Epic 2 functionalities.
- Implement any enhancements or additional features if time allows.
- Conduct final user testing and feedback sessions.
- Finalize project documentation and prepare the presentation.
- Ensure deployment and setup instructions are well-documented.

Final Deadlines

- **Project Due: May 2nd** (Ensure everything is submitted and meets requirements).
- **Presentation: May 5th** (Prepare slides, demos, and team roles for the final presentation).