Momentum: A Content Creator's All in One Productivity Cross-Platform App

Marqice A. Rector

Keiser University

CEN4090: Software Engineering Capstone

Professor Kruayatidee, Somard

August 28,202

Project Overview

Content creators face unique challenges in organizing ideas, planning content, and managing production across multiple platforms. Most existing productivity tools fail to address these specialized needs, resulting in fragmented workflows and lost efficiency.

This capstone project proposes a cross-platform productivity and planning application tailored specifically for content creators. The app enables users to capture ideas, schedule content, manage production workflows, and track key analytics—all within a unified and intuitive interface.

Developed using React Native and Firebase (React Native, n.d.; Firebase, n.d.), the solution is designed for both mobile and web use, supporting creators wherever they work. The aim is to streamline the content creation process and empower creators to consistently move from concept to publication with greater organization and less stress.

User Stories

Momentum is designed to serve the daily workflow of content creators, influencers, and digital entrepreneurs who manage multiple ideas, deadlines, and performance metrics across platforms. The following user stories illustrate real-world scenarios and motivations for using the application:

- Idea Capture & Organization:

  As a content creator, when inspiration strikes while I'm commuting or in between meetings, I want to instantly log my idea in Momentum, add a quick description, tag it with a relevant platform (e.g., YouTube, TikTok), and categorize it by theme so that I can retrieve and develop it later without losing valuable concepts.

- Content Planning & Scheduling:

As a creator preparing for a product launch, I want to assign specific tasks to different dates on the integrated calendar, such as “Film video demo” or “Publish Instagram teaser,” so that I have a clear, actionable schedule and can track my progress leading up to the launch.

- Cross-Platform Analytics Tracking:

  As a creator who posts across YouTube, TikTok, and Instagram, I want to manually log performance data for each piece of content (views, likes, comments, and postdate) and filter analytics by platform so I can identify which channels are driving the most engagement and adjust my strategy accordingly.

- Daily Dashboard Overview:

  As a busy creator, I want my dashboard to display my most urgent tasks, upcoming deadlines, and highlighted ideas so that when I log in each morning, I can immediately focus on what matters most for that day without searching through different sections.

- Content Strategy Adjustment:

  As a creator who decides to pivot my content direction mid-month, I want the ability to quickly edit or remove outdated tasks, ideas, or analytics entries so my Momentum workspace always reflects my current priorities and strategy.

- Profile Customization & Workflow Personalization:

  As a user, I want to set my preferred theme (light or dark mode), update my personal information, and adjust notification settings so that my experience feels tailored to my working style and my workspace is visually comfortable during long work sessions.

- Secure Access & Data Privacy

  As a creator managing sensitive performance data, I want secure account login and logout features so that my insights, schedules, and strategies remain private and cannot be accessed by unauthorized users.

- Campaign Execution Tracking

  As a creator running a month-long campaign, I want to track progress on related tasks, attach analytics entries to specific campaign ideas, and see how the overall project performed compared to my initial goals so that I can use the results to improve my next campaign.

Problem Statement

Content creators often manage multiple projects, platforms, and deadlines simultaneously, requiring careful coordination and organization. However, most available productivity tools are designed for general task management and lack features that address the specific workflows and requirements of content creation. As a result, creators are forced to rely on a combination of generic apps, spreadsheets, and manual reminders, leading to inefficiency, missed opportunities, and increased risk of burnout.

There is a clear need for a unified solution that caters to the distinct planning, tracking, and organizational demands of modern content creators.

Project Scope

This project will deliver a cross-platform productivity and planning application specifically designed for content creators. The core features will include:

- Idea Bank: Capture, categorize, and manage content ideas.
- Content Calendar: Visually schedule upcoming posts across multiple platforms.
- Workflow Tracker: Use customizable checklists to monitor each stage of content production.
- Manual Analytics Logging: Record and review basic performance metrics for published content.
- Reminders and Notifications: Receive alerts for deadlines and important tasks.
- User Authentication: Secure sign-up and login functionality.

- Responsive Design: Seamless experience across mobile devices and web browsers.

Out of Scope

This project will for now not include automated integration with external analytics APIs (such as YouTube or Instagram), advanced media editing capabilities, or real-time team collaboration beyond individual user accounts.

The goal is to deliver a fully functional minimum viable product (MVP) within an eight-week development timeline.

Objectives

The primary objective is to equip content creators with a unified tool that increases productivity, enhances organization, and reduces friction in the content creation process.

- Develop a cross-platform application that streamlines content planning and workflow management for content creators.
- Enable users to efficiently capture and organize content ideas in a structured and accessible format.
- Provide a visual scheduling tool that allows creators to plan, track, and manage upcoming content across multiple platforms.
- Implement customizable production workflows that guide users through each step of content creation.
- Offer manual analytics tracking to help creators monitor performance and growth.
- Deliver a responsive, user-friendly interface that works seamlessly on both mobile devices and web browsers.
- Ensure secure user authentication and data management using modern technologies and best practices.

Requirements

Functional Requirements

The functional requirements define the core features and behaviors that the application must support to meet the needs of content creators. These requirements specify the essential actions and interactions that users should be able to perform within the app:

User Authentication

- The system shall allow users to register and log in securely.
- The system shall support password reset functionality.

Idea Bank

- Users shall be able to add, edit, and delete content ideas.
- Users shall be able to categorize and tag ideas.

Content Calendar

- Users shall be able to schedule upcoming posts on a visual calendar.
- The calendar shall display scheduled content and deadlines.

Workflow Tracker

- Users shall have customizable checklists for each content piece.
- Users shall be able to update the status of content as it progresses through production stages (e.g., idea, script, record, edit, publish).

Manual Analytics Logging

- Users shall be able to manually enter performance metrics (e.g., views, likes, comments) for each published piece of content.
- The system shall display basic charts or summaries of analytics data.

Reminders and Notifications

- Users shall receive reminders for upcoming deadlines and tasks.

- Users shall be able to customize notification settings.

Responsive **Design**

- The application shall function seamlessly on both mobile and web platforms.

Non-Functional Requirements

The non-functional requirements describe the overall qualities and constraints of the application, ensuring a reliable and high-quality user experience. These requirements address aspects such as performance, security, usability, compatibility, and maintainability:

Performance:

- The application shall load all main screens within two seconds on standard hardware and network conditions.

Security:

- All user data shall be stored securely and transmitted over encrypted channels.
- The app shall comply with basic data privacy standards.

Reliability:

- The system shall maintain at least 99% uptime during the project demonstration period.

Usability:

- The interface shall be intuitive, with minimal learning curve for new users.
- Help and support information shall be accessible within the app.

Compatibility:

- The application shall support the latest versions of major browsers and mobile operating systems (iOS and Android).

Maintainability:

- The codebase shall follow standard development practices and be documented for future updates.

System Design

The content creator productivity app is designed as a cross-platform solution, leveraging a modern client–server architecture to ensure a responsive, reliable, and scalable experience for users on both mobile and web platforms. The system is divided into several main components.

The client application was developed using React Native with Expo for web compatibility (React Native, n.d.; Expo, n.d.). React Native was chosen because it allows for cross-platform development with a single codebase, significantly reducing development time and ensuring consistency across iOS, Android, and web platforms. Expo was integrated to streamline the build process and simplify access to device-specific features such as notifications, while also allowing easy deployment for both testing and production.

The backend services are managed through Firebase (Firebase, n.d.), which was selected for its scalability and reliability as a backend-as-a-service (BaaS) platform. Firebase provides user authentication, real-time data storage, and notification services out of the box, reducing the need for custom backend infrastructure and lowering the overall complexity of the project. Its native integration with React Native also ensured smooth communication between the frontend and backend.

For data management, Firebase's Firestore database was chosen as the primary solution for storing user information such as content ideas, calendar events, workflow progress, and analytics entries. Firestore is a NoSQL cloud database that offers real-time synchronization, scalability, and flexible document-based data structures. These features were particularly important for this project because users require immediate updates across devices and the ability to scale seamlessly as the application grows.

Notifications are managed using Firebase Cloud Messaging in combination with Expo Notifications. This approach was chosen to ensure reliable delivery of reminders and deadline alerts across mobile and web platforms. Firebase Cloud Messaging is a robust, industry-standard service for push notifications, while Expo Notifications simplifies integration and testing on both iOS and Android devices.

Finally, security is a key consideration across the system. All communication between the client and backend occurs over encrypted HTTPS connections, while authentication tokens are securely managed. Firebase security rules were implemented to ensure that each user can only access their own data, thereby maintaining privacy and integrity throughout the application.

Overall, the system design prioritizes modularity, scalability, and maintainability. By combining React Native, Expo, and Firebase services, the application achieves a balance between rapid development, reliable performance, and the ability to evolve with future feature demands.

Technology Stack

The app's technology stack was carefully chosen to align with project goals while minimizing development overhead. The frontend was built with React Native because of its ability to deliver a unified codebase for cross-platform development. This not only reduces redundancy in coding but also provides a familiar component-based architecture, making it easier for developers to maintain and expand the app. Expo was included in the frontend layer to handle platform-specific features, testing, and deployment in a seamless manner.

The backend relies on Firebase, which eliminates the need for custom servers while still providing robust authentication, database, and notification services. Firebase was chosen because it is cost-effective for small-scale deployments, yet highly scalable to support future growth.

For the database, Firestore was selected due to its flexible document structure and real-time data synchronization. Unlike relational databases, Firestore adapts well to evolving data requirements, making it ideal for an application that manages dynamic content such as ideas, tasks, and analytics entries.

Notifications are implemented using a hybrid of Firebase Cloud Messaging and Expo Notifications. This combination was chosen to provide reliable push notifications across multiple platforms without requiring complex, custom notification infrastructure.

Lastly, deployment was handled through Expo Go and Expo Build. This choice simplified the process of running the application on both physical devices and simulators, accelerating testing and delivery to multiple platforms simultaneously.

In summary, each technology was selected to support the project's emphasis on scalability, rapid iteration, and cross-platform compatibility, ensuring the final product meets user needs while remaining sustainable for future development.

Technology Stack Overview

Layer	**Technology**	**Purpose/Notes**
Frontend	React Native	Cross-platform UI for mobile and web (via Expo)
Frontend	Expo	Streamlines mobile/web builds, device features, and testing
Backend	Firebase	Provides backend-as-a-service, authentication, database
Database	Firestore	Cloud NoSQL database for storing user/content data

Layer	Technology	Purpose/Notes
Notifications	Expo Notifications / Firebase Cloud Messaging	Push notifications/reminders
Deployment	Expo Go / Expo Build	App deployment to iOS, Android, and web

Component/Modular Design

The Momentum Productivity App is structured around a modular architecture, where each core feature is encapsulated within its own component or screen. This design approach supports scalability, ease of maintenance, and clear separation of concerns. The following describes the major components and their primary responsibilities within the application. References to the corresponding wireframes and diagrams are provided for further clarity.

Dashboard

The dashboard serves as the central hub of the application, providing users with a quick overview of their most important information. It highlights the next three to five upcoming tasks or events, displays flagged or major ideas that need attention, and presents quick statistics such as total tasks, ideas, or productivity streaks. In addition, the dashboard provides navigation shortcuts to other sections, ensuring that users can manage their content creation workflow efficiently.

Idea Bank

The idea bank provides a dedicated space for users to capture, organize, and manage their creative concepts. Users can add, edit, and delete ideas while keeping them neatly arranged in a scrollable list sorted by date. Important ideas can be starred or flagged for focus, ensuring that no valuable concept is overlooked. This component acts as the foundation for building and refining content plans.

Calendar

The calendar component enables users to schedule, review, and track events tied to specific dates. With a visual calendar interface, users can easily select dates, view associated tasks, and add, edit, or delete events. Integration with the dashboard ensures that urgent or upcoming tasks appear prominently, helping users stay organized and on schedule.

Analytics

The analytics module provides tools for users to manually log and track the performance of their content across different platforms. Performance data such as views, likes, and comments can be entered, visualized with interactive charts, and filtered by platform or date. This allows users to gain insights into their productivity and content reach, empowering them to make informed decisions.

Settings

The settings section centralizes user preferences and account management. From here, users can update profile details, toggle between light, dark, or system themes, and manage notifications. It also includes account functions such as login, logout, and app version details. By consolidating these features, settings give users full control over personalization and account security.

Navigation

Navigation is designed to create a seamless user experience by enabling smooth transitions between the app’s core sections. It uses a bottom tab navigation bar with clear visual feedback to indicate the active section. With quick access to the dashboard, idea bank, calendar, analytics, and settings, navigation ensures users can move fluidly through the app.

Data and Backend Integration.

Data and backend services manage the app’s persistence, authentication, and synchronization. Firebase Firestore stores user data, including ideas, tasks, analytics, and settings, while Firebase

Authentication provides secure login and logout functionality. Real-time synchronization between the app and cloud ensures that user data is always up to date, creating a reliable and consistent experience.

This modular design allows each feature to be developed, tested, and updated independently, supporting a robust and scalable application architecture. For detailed UI layouts, see the respective wireframes included in the appendix. For data flow and entity relationships, refer to the ER and architecture diagrams.

Testing Plan

The Momentum application employs a structured testing approach designed to ensure functionality, reliability, and maintainability within the constraints of the capstone timeline. A test-driven development (TDD) methodology was adopted for logic-intensive components, such as analytics calculations, calendar utilities, and data mapping. TDD was chosen because it promotes early detection of defects, enforces well-defined interfaces, and creates a regression safety net as features evolve (Beck, 2003).

Testing was performed at multiple levels. Unit testing focused on validating isolated functions and business logic, implemented using **Jest** as the primary testing framework. Jest was selected for its speed, integrated mocking, and coverage reporting, which make it well-suited for React Native environments (Meta, 2023). To test user interactions, the project applied the **React Native Testing Library**, which emphasizes accessibility-based queries and behavior-driven testing. This approach was chosen because it aligns closely with user experience by evaluating outcomes visible to end users rather than internal implementation details (CallStack, 2023).

Integration testing was conducted by combining screens with mock navigation and simulated Firebase responses through the **Firebase Emulator Suite**. This ensured that workflows, such as adding tasks or filtering analytics, could be validated without depending on live data. End-to-end (E2E) testing was also incorporated using **Detox**, which automates user flows on simulators and emulators to validate navigation, authentication, and task management in realistic conditions (Wix, 2023). Finally, **user acceptance testing (UAT)** was performed with peers to confirm usability and alignment with project requirements.

To automate these processes, the team configured a Jenkins continuous integration (CI) pipeline. Jenkins was selected because it provides a consistent environment for running tests on every commit and prevents regressions from being merged into the codebase (Jenkins, 2023). The pipeline executes linting, unit tests, integration tests, and E2E scenarios against emulator services. Coverage thresholds were enforced to maintain quality, and build reports were generated for transparency.

This testing strategy balances strict TDD for high-value modules with pragmatic UI and E2E testing for broader coverage, thereby ensuring that Momentum meets quality expectations without exceeding development resources.

Implementation Plan/ Schedule

This project will be developed and delivered over an eight-week period. The implementation is organized into phases to ensure steady progress and timely completion of all core features.

Week-by-Week Schedule:

Week 1: Planning & Setup

- Finalize project requirements and technical specifications
- Set up project repositories and development environment
- Create initial mockups and wireframes

Weeks 2–3: Core Functionality Development

- Implement user authentication (sign up, login, password reset)
- Build the idea bank for adding, editing, and deleting content ideas
- Develop basic UI components and layouts

Week 4: Content Calendar & Workflow

- Develop the content calendar for scheduling and viewing planned content
- Implement customizable workflow checklists for content production stages

Week 5: Analytics & Notifications

- Enable manual analytics logging and display basic performance summaries
- Integrate reminder and notification features

Week 6: Responsive Design & Cross-Platform Testing

- Refine UI for consistent experience across mobile and web
- Conduct testing and debugging on different devices and browsers

Week 7: Polish & Documentation

- Address remaining bugs and optimize performance
- Finalize project documentation, including user guide and system notes

Week 8: Final Testing & Presentation Preparation

- Conduct comprehensive testing and quality assurance
- Prepare project presentation and deployment for demonstration

Milestones:

- Core features (authentication, idea bank) functional by end of Week 3
- Calendar and workflow features integrated by end of Week 4
- Full MVP completed and tested by end of Week 6
- Documentation, polish, and presentation-ready deployment by end of Week 8

Conclusions & Limitations

The development of *Momentum* demonstrates the feasibility and value of a dedicated productivity application tailored to the unique needs of content creators. By combining idea management, scheduling, and manual analytics tracking into a single platform, the app addresses common challenges faced by individuals who juggle multiple creative projects across different platforms.

The modular architecture and integration with Firebase provide a foundation that is both scalable and adaptable for future growth.

However, several limitations emerged during the design and development process due to time, resource, and technical constraints. One of the most significant is the absence of automated analytics integration with external platforms such as YouTube, TikTok, or Instagram. While users can still track performance metrics by manually inputting data, this process may be cumbersome and less accurate compared to automated solutions. Real-time data collection through APIs would provide more seamless insights and allow creators to make quicker, evidence-based decisions.

Another limitation lies in the notification system, which currently relies on Firebase Cloud Messaging and Expo Notifications. Although functional, the system may not yet be fully optimized for advanced scheduling features or personalized reminders. Similarly, cross-platform testing was conducted within limited device environments, which means additional usability testing across a broader range of devices and screen sizes would be necessary to guarantee a universally consistent user experience.

Lastly, the initial version of the app emphasizes core functionality over advanced features. Capabilities such as workflow automation, collaborative content planning, or advanced data visualization tools were intentionally deferred to future iterations. These features, once added, could significantly enhance the app's ability to empower creators and streamline their processes.

In conclusion, while *Momentum* succeeds in addressing its central goals and demonstrates strong potential as a productivity tool, it remains an evolving product. The limitations identified in this stage of development highlight areas of opportunity for future research and enhancement, ensuring that subsequent versions can offer even greater value to its target users.

References

Addison-Wesley. (2003). *Test-driven development: By example* (K. Beck). Addison-Wesley.

CallStack. (2023). *React Native Testing Library documentation*. https://testing-library.com/docs/react-native-testing-library/intro

Detox. (2023). *Detox end-to-end testing and automation for mobile apps*. Wix. https://wix.github.io/Detox/

Firebase. (2023). *Firebase Emulator Suite*. Google. https://firebase.google.com/docs/emulator-suite

Firebase. (2023). *Firebase documentation*. Google. https://firebase.google.com/docs

Jenkins. (2023). *Jenkins user documentation*. https://www.jenkins.io/doc/

Meta. (2023). *Jest: Delightful JavaScript testing*. https://jestjs.io/

Meta. (2023). *React Native: Create native apps for Android and iOS using React*. https://reactnative.dev/

NPM, Inc. (2023). *Node.js JavaScript runtime*. https://nodejs.org

OpenAI. (2023). *Expo: An open-source platform for making universal native apps*. https://expo.dev/

React Navigation. (2023). *React Navigation documentation*. https://reactnavigation.org/

Trello. (2023). *Trello project management tool*. Atlassian. https://trello.com

GitHub, Inc. (2023). *GitHub Issues documentation*. https://docs.github.com/issues