**The Annoying Alarm**

**Karan Dhar**

**Danson Wachira**

**Bao Bui**

**Joseph Downs**

**Continuous Integration and Deployment Plan for The Annoying Alarm**

The **Continuous Integration and Deployment (CI/CD) plan** for *The Annoying Alarm* focuses on streamlining the development, testing, and deployment processes to ensure a reliable and scalable application. The goal of this plan is to minimize errors, enhance collaboration, and speed up feature delivery without compromising quality. Below is the detailed CI/CD process tailored to *The Annoying Alarm*.

**Continuous Integration (CI) Process**

Continuous Integration ensures that any changes made to the application are integrated into a shared codebase regularly, tested automatically, and validated to catch errors early. For *The Annoying Alarm*, this process begins whenever a developer submits changes to the code repository.

1. **Code Integration and Version Control**: The project relies on a Git-based version control system, such as GitHub or GitLab, where every feature or bug fix is developed in a separate branch. When a developer pushes changes, a pull request triggers the CI pipeline. This approach ensures that no code enters the main branch without being thoroughly reviewed and tested.
2. **Automated Testing**: Automated tests play a crucial role in the CI process. The pipeline runs a suite of unit tests, integration tests, and regression tests to verify that:

Alarm features (e.g., setting, snoozing, and stopping alarms) work as intended. Puzzles are triggered correctly and completed actions stop the alarm. System-wide behaviors, such as volume adjustments and background operations, function correctly. These tests ensure that new code does not break existing functionality and meet the defined pass criteria, such as 100% reliability of alarm triggering and puzzle functionality.

1. **Build Validation**: Once tests pass, the application is built into an executable format (e.g., an Android APK file). The pipeline ensures that the build is successful on various configurations and environments. For *The Annoying Alarm*, builds must be compatible across different Android versions and device types.
2. **Error Reporting and Feedback**: If tests or builds fail, the CI system automatically notifies developers via email or chat tools like Slack. This rapid feedback allows developers to address issues immediately, maintaining smooth development progress.

**Continuous Deployment (CD) Process**

The deployment phase ensures that the tested code is released to staging or production environments automatically and with minimal downtime. For *The Annoying Alarm*, the deployment process can be automated to deliver updates to beta users or directly to app stores.

1. **Staging Environment Deployment**: After passing CI, the application is deployed to a staging environment that mirrors the production setup. This environment is used for additional manual testing, such as checking the app's behavior when the device is locked, testing edge cases in alarm triggering, and verifying puzzles under different conditions.
2. **User Acceptance and Beta Testing**: Updates to *The Annoying Alarm* may first be released to a group of beta testers. This step gathers feedback on new features or bug fixes before deploying to all users. During this stage, automated deployment tools such as Firebase App Distribution or Google Play's Beta Track are used to distribute the app seamlessly to testers.
3. **Production Deployment**: After successful testing in the staging environment, the application is deployed to the production environment, such as the Google Play Store. Deployment is handled using automated scripts or tools to eliminate manual errors. For instance: o Incrementing the app version and updating release notes are automated to ensure consistency. o Gradual rollouts can be configured to monitor the app’s performance on a small percentage of users before making it available to everyone.
4. **Monitoring and Rollback Mechanisms**: Continuous monitoring is essential after deployment to detect crashes, bugs, or performance issues. Tools like Crashlytics or Google Play Console's ANR/crash reports are used to identify issues in real time. If a critical issue arises, the deployment system includes a rollback mechanism to revert to a stable version automatically.

**Advantages of CI/CD for The Annoying Alarm**

* **Improved Quality**: Automated testing ensures that bugs are caught early, reducing the risk of releasing faulty features to users.
* **Faster Delivery**: Automation of builds, tests, and deployments enables quicker delivery of updates and new features, such as additional puzzles or improved alarm functionalities.
* **Scalability**: The modular architecture of *The Annoying Alarm* integrates seamlessly with CI/CD pipelines, making it easier to add new features, such as multiple alarms or dynamic puzzle difficulty.
* **Consistency**: With automated deployment processes, releases are consistent and error-free, ensuring users receive reliable updates.

By implementing a robust CI/CD pipeline, *The Annoying Alarm* achieves a streamlined workflow for developing, testing, and releasing features. This process ensures that the app remains functional, scalable, and user-friendly, even as its complexity grows.

**Testing and Quality Assurance for *The Annoying Alarm***

Testing and quality assurance for *The Annoying Alarm* focus on ensuring that the application operates reliably and meets its functional and performance requirements. The strategy involves creating a robust testing scheme, realistic testing samples, and well-defined pass/fail criteria to ensure the app performs as expected under various conditions.

**Testing Scheme**

The testing scheme combines **unit testing** and **integration testing** to validate individual components and the interaction between features. Each test focuses on critical functionalities like alarm triggering, snoozing, puzzle engagement, and overall app performance.

* **Unit Testing**: Each feature is tested in isolation to confirm its behavior. For instance, the alarm-creation function, snooze mechanism, puzzle initiation, and alarm stopping are tested independently. These tests ensure that individual parts of the app function correctly.
* **Integration Testing**: These tests ensure that different components work seamlessly together. Key workflows, such as the alarm ringing at the scheduled time, initiating a puzzle when snoozing or stopping the alarm, and handling volume adjustments during puzzle interaction, are tested to confirm the app's overall functionality.

This scheme ensures that both isolated features and their interactions align with the expected user experience.

**Testing Samples**

Testing samples simulate realistic scenarios and user interactions to validate the app under conditions it will encounter in real use:

1. **Alarm Configurations**: A set of sample alarms is created with varying schedules, such as alarms set a few minutes apart or across different times of the day. This ensures that alarms trigger reliably and on time under diverse settings.
2. **Puzzle Functionality**: Sample puzzles, such as simple math problems or pattern matching, are integrated and tested to verify their behavior when users interact with the alarm. These puzzles simulate the real challenge users must complete to stop or snooze the alarm.
3. **Device States**: The app is tested under various device states, such as when the app is running in the foreground, when it is closed, or when the device is locked. This ensures the alarm behaves consistently regardless of the device state or background conditions.

By using realistic testing samples, the team can validate that the app performs correctly in real-world scenarios.

**Design and Explanation of Software Tests**

Each test is designed to ensure specific functionalities work as intended. Below are the primary tests conducted:

1. **Alarm Trigger Test**: This test ensures alarms trigger at the scheduled time with 100% reliability. For example, alarms set a few minutes apart are checked to confirm that they trigger accurately, even when the app is closed or the device is locked.
2. **Snooze Functionality Test**: This test simulates a user snoozing the alarm, verifying that it rings again after the preset snooze interval. The test ensures that the snooze function operates reliably and without delay.
3. **Puzzle Trigger and Completion Test**: The test verifies that the puzzle view appears whenever the user attempts to snooze or stop the alarm. It also ensures that solving the puzzle successfully stops the alarm sound. For instance, the alarm should stop ringing immediately upon solving a math problem or completing a pattern puzzle.
4. **Volume Adjustment Test**: This test validates the dynamic volume reduction feature, ensuring that the alarm's volume decreases when the user begins solving the puzzle. It confirms that this adjustment enhances the user experience while maintaining functionality.

These tests cover both isolated functionalities and end-to-end workflows, ensuring the app behaves as intended in all scenarios.

**Test Objectives and Metrics**

Each test has clear objectives and measurable pass/fail criteria to determine the app's reliability and performance:

1. **Alarm Triggering**:
   * **Objective**: The alarm must ring at the scheduled time, even when the app is closed or the device is locked.
   * **Pass Criteria**: The alarm triggers 100% of the time without fail.
   * **Fail Criteria**: The alarm does not trigger at the scheduled time or is delayed.
2. **Snooze Functionality**:
   * **Objective**: The alarm should ring again within the specified snooze interval. o **Pass Criteria**: The alarm reactivates consistently within the expected timeframe.
   * **Fail Criteria**: The alarm fails to reactivate or reactivates outside the acceptable time range.
3. **Puzzle Engagement**:
   * **Objective**: The puzzle must appear each time the user interacts with the alarm's snooze or stop options, and completing it must stop the alarm.
   * **Pass Criteria**: The puzzle screen appears reliably, and solving the puzzle stops the alarm immediately. o **Fail Criteria**: The puzzle does not appear, cannot be completed, or fails to stop the alarm after completion.
4. **Volume Adjustment**:
   * **Objective**: The alarm volume should decrease dynamically when the user begins solving a puzzle.
   * **Pass Criteria**: The volume reduces immediately when the puzzle screen is displayed and remains at a lower level during puzzle interaction.
   * **Fail Criteria**: The volume does not adjust, or the adjustment causes delays or other performance issues.

By following this structured testing and quality assurance approach, *The Annoying Alarm* ensures its core functionalities, such as alarm triggering, puzzle engagement, and snoozing, meet user expectations under real-world conditions. This process not only validates the app's reliability but also provides a framework for scaling and enhancing features in the future.

**Documentation and User Manual for *The Annoying Alarm***

Do you often find yourself sleeping through alarms, missing important appointments, or giving in to the temptation of the snooze button? Do you enjoy starting your day with a bit of mental stimulation? If so, *The Annoying Alarm* is your perfect solution! Unlike traditional alarms that can be silenced with a simple tap, *The Annoying Alarm* requires users to complete engaging puzzles or minigames before they can snooze or turn it off. Designed to wake you up while sharpening your mind, *The Annoying Alarm* ensures you start your day alert and energized. This documentation serves as a guide for both users and developers, providing clear instructions and technical insights for effective use and future development. The app allows you to set alarms like any standard alarm app, but the twist is that you cannot snooze or turn them off until you complete a puzzle or minigame, such as a sliding puzzle to rearrange tiles into a picture, an addition puzzle to solve a quick math problem, or an RGB Guru puzzle to match colors or solve logic-based color sequences. To get started, download *The Annoying Alarm* from the App Store or Google Play and follow the prompts to install it on your device. Set your desired alarm time, customize the ringtone (coming soon!), and select a puzzle or enable random puzzle assignment for variety. Adjust puzzle difficulty based on your preference, enable dynamic difficulty for more challenges if you snooze repeatedly, and manage multiple alarms efficiently through the intuitive interface. For troubleshooting, ensure the app has necessary permissions like notifications and background activity if the alarm does not trigger, verify your device’s media and alarm volumes if the volume malfunctions, and check that a puzzle type is assigned to the alarm if puzzles fail to appear. Optimize your experience by experimenting with different puzzles, scheduling alarms with staggered minigames to ensure full alertness, and utilizing dynamic difficulty for gradual increases in puzzle challenges. For developers, *The Annoying Alarm* employs a modular architecture to facilitate easy maintenance and future development. Key components include the Alarm Scheduler, which handles alarm timing, the Puzzle Engine, which initiates assigned puzzles or minigames, and the Snooze Controller, which manages snooze functionality and integrates with the dynamic difficulty system. The app’s core workflow transitions seamlessly from alarm activation to puzzle completion verification before allowing snooze or dismissal. The modular design ensures each component operates independently, enabling future enhancements without disrupting existing functionality. Planned features include new puzzles to expand the library, custom ringtones for users to select their preferred alarm sounds, dynamic difficulty to automatically increase puzzle complexity with frequent snoozes, and puzzle preferences to let users choose their favorite minigames. Development insights emphasize scalability, cross-platform compatibility for Android and iOS, and the incorporation of user feedback for continuous improvement. *The Annoying Alarm* combines functionality, fun, and innovation to revolutionize how you wake up, providing everything needed for end-users to conquer the snooze button and for developers to enhance the app. Start your day the right way with *The Annoying Alarm*!

**Progress Reports for *The Annoying Alarm***

The development of *The Annoying Alarm* is now complete, with all core functionalities fully implemented and tested. The alarm system operates reliably, including essential features like alarm creation, snooze functionality, and puzzle-based dismissal. Extensive testing has confirmed the app performs seamlessly in various scenarios, whether the app is running in the background or completely closed. The puzzle library, a key component of the app’s uniqueness, has been successfully finalized. Users can now enjoy the following puzzles:

* **Sliding Puzzle:** Rearrange tiles to form a complete image.
* **RGB Guru Puzzle:** Match a given color by manipulating the Red, Green, and Blue sliders.
* **Addition Puzzle:** Solve a math equation to turn off the alarm. With these engaging and diverse puzzles, the app ensures a stimulating and effective wake-up experience.

The project is now ready for broader testing and user feedback, bringing it closer to full-scale deployment.

**Meeting Minutes for *The Annoying Alarm***

For *The Annoying Alarm*, meetings are held periodically to ensure smooth progress and resolve any roadblocks. These meetings take place either online or offline, depending on the availability and preferences of the team members. The flexibility of this approach ensures that team members can contribute effectively regardless of their location or schedule. When urgent assistance or clarification is required, meetings are promptly organized to address specific concerns or issues.

The primary platform for communication and collaboration is Discord, where the team remains highly active. Discord serves as the main hub for sharing updates, discussing ideas, and troubleshooting challenges. This constant communication ensures that all team members are aligned and informed about the project's progress.

Meeting discussions typically revolve around task updates, testing results, and the integration of new puzzles into the app. Action items are assigned during these meetings, with clear responsibilities and deadlines to maintain momentum. By maintaining an open and collaborative environment, the team ensures that any challenges are quickly identified and resolved, keeping the project on track.

The informal yet effective meeting structure, combined with active Discord communication, allows the team to work efficiently and collaboratively, ensuring the successful development of *The Annoying Alarm*.

**Security Considerations for The Annoying Alarm**

**Potential Vulnerabilities**

1. **Data Privacy Concerns**:
   * If the app collects user data, such as alarm schedules or device configurations, this data could be at risk of unauthorized access if not secured properly.

Storing sensitive information locally on the device without encryption can make it vulnerable to attacks.

1. **API and Backend Vulnerabilities**:
   * If the app communicates with backend servers (e.g., for analytics or updates), these APIs may be susceptible to common threats like SQL injection, insecure authentication, or data interception.
2. **Unauthorized Access**:
   * Malicious actors could exploit weak authentication mechanisms to tamper with app settings, disable alarms, or access sensitive features.
3. **Code Vulnerabilities**:
   * Hardcoded secrets, poorly written code, or unvalidated input may leave the app open to attacks such as reverse engineering or buffer overflow.

**Continuous Security Assessment Plan**

To maintain robust security, the following steps can be implemented as part of a continuous security assessment strategy:

1. **Static Application Security Testing (SAST)**:
   * Regularly run tools like SonarQube or Checkmarx to scan the codebase for vulnerabilities such as hardcoded credentials or unsafe functions. o Include these scans in the CI/CD pipeline to ensure every update is reviewed for security risks.
2. **Dynamic Application Security Testing (DAST)**:
   * Simulate real-world attacks on the app (e.g., penetration testing) to identify vulnerabilities in the app's runtime environment and APIs.
3. **Secure Data Practices**:
   * Encrypt all sensitive data stored locally or transmitted over networks using industrystandard encryption protocols such as AES-256 or HTTPS.
   * Implement secure data erasure practices to prevent leaks when uninstalling the app.
4. **Regular Dependency Audits**:
   * Use dependency scanning tools like OWASP Dependency-Check to ensure third-party libraries used in the app are up-to-date and free of known vulnerabilities.
5. **User Authentication and Role Management**:
   * Integrate secure authentication mechanisms, such as biometric authentication or two factor authentication, to minimize unauthorized access.

**Scalability and Performance**

**Software Design for Scalability and Performance**

1. **Modular Architecture**:
   * The app already uses a modular architecture separating alarm scheduling, puzzle initiation, and snooze functionalities. This enables easy scaling by allowing individual modules to be updated or extended independently.
2. **Efficient Resource Usage**:
   * **Alarms**: Use lightweight system services for triggering alarms, reducing battery and CPU usage.
   * **Puzzles**: Optimize puzzle rendering and logic to ensure they do not consume excessive memory, especially on lower-end devices.
3. **Load Balancing for Backend Services**:
   * If the app employs backend services, consider implementing load balancers to distribute traffic evenly, ensuring smooth operation during peak usage.
4. **Caching Strategies**:
   * Cache static assets and frequently used configurations locally to reduce latency and reliance on network operations.
5. **Testing for Scalability**:
   * Perform stress and load testing using tools like JMeter or Apache Benchmark to simulate high user volumes and measure the app's performance under these conditions. 6. **Economical and Time Constraints**:
   * Focus on using free or affordable solutions such as Firebase for backend services, which offer scalability out of the box for small teams and startups.
   * Prioritize features that impact user experience (e.g., alarm reliability) over less critical functionalities.

**Legal and Compliance Considerations**

**Societal Constraints and Standards**

1. **User Data Protection**:
   * Comply with data protection regulations like GDPR (General Data Protection Regulation) or CCPA (California Consumer Privacy Act), especially if the app stores or processes personal data.

Ensure users have full control over their data, such as the ability to delete alarms or reset app data.

1. **Accessibility Standards**:
   * Design the app to be accessible to users with disabilities by implementing features like high-contrast modes, voice commands, or compatibility with screen readers.
2. **Licensing of Dependencies**:
   * Ensure all third-party libraries and APIs used in the app comply with open-source licensing requirements to avoid legal disputes.
3. **Age-Appropriate Content**:
   * Since the app uses puzzles, ensure the content is appropriate for all age groups or provide options for customization based on user preferences.

**Conclusion**

The Annoying Alarm has successfully addressed key challenges in the alarm application space by prioritizing security, scalability, performance, and legal compliance. With robust security measures in place, the application ensures that user data remains protected, while its scalable architecture is ready to handle increasing demand as it grows. Designed for optimal performance, it offers a seamless user experience, delivering the functionality users need without lag. By adhering to legal standards, the app provides peace of mind, knowing that all regulatory requirements have been met.

The integration of CI/CD pipelines throughout the development lifecycle has ensured that each update is smooth, reliable, and in line with best practices, positioning The Annoying Alarm for continued success. This strong foundation enables the app to evolve and adapt to future user needs, whether through new features or performance enhancements.

**Why Choose The Annoying Alarm?**

* **Innovative Solution**: It offers a unique blend of functionality and engagement, turning a simple alarm into an interactive experience.
* **Boosts Productivity**: Designed to help users wake up on time and start their day off right, The Annoying Alarm acts as both a wake-up tool and a productivity booster.
* **Highly Customizable**: From alarm sounds to puzzle challenges, The Annoying Alarm is fully customizable, allowing users to tailor the experience to their preferences, making each interaction personal and motivating.

With its complete feature set and commitment to quality, The Annoying Alarm is not just another alarm app it's a comprehensive solution designed to enhance daily routines, providing users with the tools they need to start their day energized and on schedule.

Appendix:

Presentation Video: <https://youtu.be/uEw_TFJWeFo>

GitHub repository: <https://github.com/Joe-Downs/annoying-alarm>