**SDG6 Mobile App Development Documentation**

* The SDG6 Mobile App is built to help users track their daily water usage, learn about rainwater harvesting, and access offline tips related to water conservation. The app is developed using Android Studio with Firebase Realtime Database for backend services. This documentation covers the processes of version control, CI/CD, quality assurance, and production deployment to ensure a smooth development lifecycle.

**Phase 1: Version Control & Collaboration**

**1. GitHub Issues**

* To maintain clear project management and task allocation, GitHub Issues are used to track each feature, bug, and task. Create a new issue for each task to ensure visibility and accountability.

**Steps to Create Issues:**

1. Go to your GitHub repository.
2. Navigate to the "**Issues**" tab → Click "**New Issue**".
3. Create individual issues for tasks such as:
   * **"Create login screen**"
   * **"Implement Firebase authentication"**
   * **"Design daily water consumption UI"**
   * **"Integrate Firebase Realtime Database"**

**2. GitHub Project Board**

To better organize development tasks, set up a project board.

**Steps to Set Up:**

1. Go to the "**Projects"** tab in the GitHub repository.
2. Click "**New Project**" and select "Board".
3. Name the board: **SDG6 Mobile App DevOps Board**.
4. Add columns: **To Do → In Progress → Done.**
5. Link the created issues to this board for better tracking.

* **Purpose**: Visualizes development progress and task management.

**3. Git Branching Workflow**

This process ensures effective collaboration with version control.

**Steps to Set Up:**

1. Clone your repository:

“ **git clone https://github.com/Saheela1023/DevOps-App.git”**

**“cd DevOps-App”**

1. Create a dev branch:

**git checkout -b dev**

**git push origin dev**

1. For each new feature, create a feature branch:

**git checkout -b feature/login-screen**

* **Purpose**: Ensures that developers can work concurrently on different features without affecting the main branch.

**4. Pull Request Template**

To ensure consistent pull request creation, add a template.

**Steps to Add PR Template:**

1. In the GitHub repository, create the folder: github/.
2. Inside. github/, create a file: PULL\_REQUEST\_TEMPLATE.md.
3. Paste the following into the file:

**## 📝 Summary**

**Briefly explain what your PR does.**

**## ✅ Checklist**

**- [ ] Feature implemented**

**- [ ] Code builds without errors**

**- [ ] Tests added or updated**

**- [ ] Related documentation updated**

* **Purpose**: Standardizes pull requests to ensure they meet project requirements.

**5. Contributing Guide**

Provide guidelines for contributing to the repository.

**Steps to Add Contributing Guide:**

1. In the root of your repo, create a file: CONTRIBUTING.md.
2. Paste the following into the file:

**# 🚀 Contributing Guide**

**## Git Workflow**

**- Use the dev branch for active development.**

**- Create a feature branch: feature/your-feature-name**

**- Use pull requests to merge into dev.**

**## Commit Message Format**

**Follow Conventional Commits:**

**- feat: for new features**

**- fix: for bug fixes**

**- docs: for documentation**

* **Purpose**: Provides clear instructions on how to contribute and maintain consistency across contributions.

**Phase 2: Continuous Integration/Continuous Delivery (CI/CD)**

**6. Set Up GitHub Actions for Android App CI/CD**

Automate the process of building and testing your Android app using GitHub Actions.

**Steps to Set Up CI/CD:**

1. Create the folder: github/workflows/.
2. Inside .github/workflows/, create the file: android.yml
3. Paste the following into the file:

**name: Android CI**

**on:**

**push:**

**branches:**

**- dev**

**pull\_request :**

**branches:**

**- dev**

**jobs:**

**build:**

**runs-on: ubuntu-latest**

**steps:**

**- name: Checkout code**

**uses: actions/checkout@v3**

**- name: Set up JDK**

**uses: actions/setup-java@v3**

**with:**

**java-version: '11'**

**- name: Build project**

**run: ./gradlew build**

* **Purpose**: Automates app builds and ensures that the code always passes build checks.

**7. Update README.md**

Keep the README.md file up-to-date to clearly describe the app and the technologies used.

**Example of Updated README.md:**

**## 📱 SDG6 Mobile App – database**

**This mobile app helps users track daily water usage, learn about rainwater harvesting, and access offline tips.**

**### Built With:**

**- Android Studio (Java)**

**- Firebase Realtime DB**

**- GitHub CI/CD**

* **Purpose**: Helps users and evaluators understand the app’s functionality and technologies used.

**8. Track and Close Issues as You Progress**

As you work on features, move cards on your GitHub Project Board (To Do → In Progress → Done) and close issues when tasks are complete.

* **Purpose**: Provides structured development progress tracking.

**Phase 3: CI/CD & Quality Assurance**

**9. Matrix Testing (Optional but Advanced)**

Matrix testing can help test multiple environments for compatibility (e.g., different versions of Java or Android SDK).

**10. Add Instrumented Tests (Android)**

Instrumented tests help you test your app’s UI on actual devices.

**Steps to Add Instrumented Tests:**

1. Open your app in Android Studio.
2. Navigate to src/androidTest/java/your\_package/.
3. Add a simple test:

**@RunWith(AndroidJUnit4.class)**

**public class ExampleInstrumentedTest {**

**@Test**

**public void useAppContext() {**

**Context appContext = InstrumentationRegistry.getInstrumentation().getTargetContext();**

**assertEquals("com.your.package.name", appContext.getPackageName());**

**}**

**}**

* **Purpose**: Verifies that the app's features work as expected on real devices.

**11. Add Security Scans (SAST & Dependency Scanning)**

**A. Dependency Scanning with GitHub Dependabot:**

Enable automatic dependency updates to prevent vulnerabilities.

* Navigate to **Settings** → **Security** → **Code security and analysis** → Enable **Dependabot alerts**.

**B. Static Analysis with CodeQL:**

1. Create a workflow in .github/workflows/codeql-analysis.yml.
2. Paste the following:

**name: "CodeQL"**

**on:**

**push:**

**branches: [dev]**

**pull\_request:**

**branches: [dev]**

**schedule:**

**- cron: '0 0 \* \* 0' # weekly**

**jobs:**

**analyze:**

**name: Analyze**

**runs-on: ubuntu-latest**

**permissions:**

**actions: read**

**contents: read**

**security-events: write**

**strategy:**

**fail-fast: false**

**matrix:**

**language: [ 'java' ]**

**steps:**

**- name: Checkout repository**

**uses: actions/checkout@v3**

**- name: Initialize CodeQL**

**uses: github/codeql-action/init@v2**

**with:**

**languages: ${{ matrix.language }}**

**- name: Autobuild**

**uses: github/codeql-action/autobuild@v2**

**- name: Perform CodeQL Analysis**

**uses: github/codeql-action/analyze@v2**

**Purpose**: Detects security vulnerabilities and coding errors in the codebase.

**Phase 4: Production Deployment & Observability**

**12. Release Strategy**

Tag releases using Git tags and draft a release on GitHub.

**Steps to Tag and Release:**

1. Tag the release:

**git tag -a v1.0.0 -m "Initial Release"**

**git push origin v1.0.0**

1. Go to **Releases** → **Draft new release** → Link the tag.

* **Purpose**: Tags your application version for deployment and releases.

**13. Set Up Observability**

Implement Firebase Crashlytics and Google Analytics for monitoring app performance.

* **Crashlytics**: Helps monitor and report app crashes.
* **Google Analytics**: Provides insights into user behavior and app events.

**14. Add Incident Response Playbook**

Create a document (docs/incident-playbook.md) that includes:

* Common issues
* Troubleshooting steps
* Contact details for team members
* **Purpose**: Helps the team quickly respond to incidents and issues.

**Conclusion**

* With this documentation, you’ve laid out clear guidelines for version control, CI/CD, testing, security, and deployment of your SDG6 Mobile App. Each step focuses on ensuring a smooth workflow, high-quality code, and seamless production deployment.