AdWatch Rewards App - Detailed Documentation

1. Introduction

AdWatch Rewards is a mobile application designed to allow users to watch ads in exchange for rewards. The app integrates various features such as a reward system, a referral program, a leaderboard, and a community section where users can interact by posting questions and comments. The backend is built using Spring Boot, and the frontend is developed in Android Studio with Kotlin.

2. Features

2.1 Ad-Watching Rewards

- Users can watch up to **30 ads per day**.
- A popup appears before watching an ad, asking for confirmation.
- Once the ad is completed, rewards are credited.
- Users can track the number of ads watched for the day.

2.2 Community Features

- Users can post **questions** and **comments**.
- Ability to **like/dislike** questions and comments.
- **Flagging system** to report inappropriate content.
- Users can **block/unblock** other users from commenting on their questions.
- The **Popular Questions** section does not load comments initially but fetches them when a question is clicked.

2.3 Referral System

- Users can **invite friends** to join the app.
- Rewards are provided when a referred user **successfully registers**.

2.4 Leaderboard

- Displays **top users** based on ad views and earnings.
- Updates dynamically.

2.5 Fraud Prevention

- **Server-Side Verification (SSV)** is implemented for ad views.
- **Duplicate IP and device detection** to prevent multiple rewards for the same user.
- Reward redemptions require additional **verification layers**.

2.6 Admin Panel

- View and manage **all users**.
- Manage **flagged questions** and **flagged comments**.
- View **blocked users**.

3. Technology Stack

Backend

- **Spring Boot** (Java)
- **Spring Security** (Authentication & Authorization)
- **Hibernate** (ORM for Database Access)
- **MySQL** (Database)
- **AdMob SSV** (Ad Fraud Prevention)

Frontend

- **Android Studio** (Kotlin)
- **Retrofit** (API Calls)
- **RecyclerView** (For displaying questions, users, etc.)

4. API Endpoints

4.1 User Authentication

- POST /register – Register a new user
- POST /login – Authenticate user

4.2 Ad System

- GET /ads/daily-limit – Fetch user's remaining ad views for the day
- POST /ads/reward – Validate and credit reward after ad completion

4.3 Community System

- POST /questions – Create a new question
- GET /questions – Fetch all questions
- POST /comments – Add a comment to a question
- POST /flag – Flag a question/comment

4.4 Referral System

- POST /referral – Register a referral
- GET /referral/status – Check referral rewards

4.5 Leaderboard

- GET /leaderboard – Fetch top users based on ad views

4.6 Admin Panel

- GET /admin/users – Retrieve all users
- GET /admin/flagged/questions – Get flagged questions
- GET /admin/flagged/comments – Get flagged comments
- GET /admin/blocked-users – Fetch all blocked users

5. Database Structure

- User Table
- Question Table
- Comment Table
- Referral Table
- Reward Table
- Transaction Table
- Flag Table
- BlockedUser Table
- LikeDislike Table
- PaymentDetails Table

6. Deployment Strategy

6.1 Backend Deployment

- Use **AWS** (EC2, RDS for MySQL)
- Deploy using **Docker Containers**
- Configure **AdMob SSV** for ad fraud detection

6.2 Frontend Deployment

- Distribute via **Google Play Store**
- Enable **Firebase Crashlytics** for error tracking

7. Security Measures

- **Encryption** for sensitive data
- **Rate Limiting** to prevent abuse

8. Future Enhancements

- **Multi-language Support**
- **AI-based Fraud Detection**
- **More Ad Networks Integration**
- **Daily Challenges & Streak Bonuses**

9. Conclusion

AdWatch Rewards is a scalable and secure application for incentivizing ad watching. With fraud prevention, a robust referral system, and an engaging community, it provides a rewarding experience for users. The backend ensures seamless transactions, while the frontend offers an intuitive interface. Future improvements will focus on expanding the platform and enhancing user engagement.

AdWatch Rewards Application Structure

The **AdWatch Rewards** application is structured using a well-organized package hierarchy, ensuring maintainability, scalability, and ease of development. Below is an overview of the application's directory structure along with the purpose of each package and file.

---

1. Root Package: com.AdWatch.AdWatch.Rewards.App

This package contains the main application entry point and various sub-packages to handle different functionalities.

- **AdWatchRewardsAppApplication.java**: This is the main entry point of the Spring Boot application.

---

2. Configuration Package (config)

This package contains configuration-related files.

- FileStorageConfig.java – Configures file storage properties.
- RedisConfig.java – Configures Redis for caching and session management.
- WebConfig.java – Configures CORS, interceptors, and other web-related settings.

---

3. Controller Package (controller)

This package handles HTTP requests and routes them to appropriate services.

- AdController.java – Handles ad-related requests.
- AdminController.java – Manages admin functionalities.
- BlockController.java – Handles user blocking/unblocking.
- CommentController.java – Manages user comments.
- FlagController.java – Manages flagging of questions and comments.
- PopularController.java – Handles popular questions and content.
- QuestionController.java – Manages questions.

- ReferralController.java – Handles referral-related operations.
- UserController.java – Manages user authentication and profile.
- VerificationController.java – Handles verification processes.
- WalletController.java – Manages user wallets and transactions.

---

4. Data Transfer Objects (DTO) Package (Dto)

This package contains DTO classes for transferring data between layers.

Includes:

- UserDTO.java, UserLoginDTO.java, UserProfileDTO.java – Manage user-related data.
- QuestionDTO.java, CommentDTO.java – Handle questions and comments.
- ReferralRequestDTO.java, ReferralResponseDTO.java – Manage referrals.
- RewardDto.java, TransactionDTO.java – Handle reward and transaction data.
- FlaggedQuestionDTO.java, FlaggedCommentDTO.java – Manage flagged content.

---

5. Entity Package (entity)

This package contains entity classes representing database tables.

- User.java, Question.java, Comment.java – Represent users, questions, and comments.
- Referral.java, Reward.java, Transaction.java – Represent referral, reward, and transaction data.
- Flag.java, BlockedUser.java, LikeDislike.java – Manage flags, blocked users, and likes/dislikes.
- PaymentDetails.java – Manages user payments.

---

6. Repository Package (repository)

This package contains interfaces extending Spring Data JPA repositories for database interactions.

Includes:

- UserRepository.java, AdRepository.java, QuestionRepository.java, CommentRepository.java
- ReferralRepository.java, RewardRepository.java, TransactionRepository.java

- FlagRepository.java, BlockedUserRepository.java, LikeDislikeRepository.java

---

7. Service Layer (service and serviceImpl)

This package contains service interfaces and their implementations.

- **Service Interface (service)**: Defines business logic methods.
  - UserService.java, AdService.java, QuestionService.java, ReferralService.java
  - WalletService.java, CommentService.java, FlagService.java, BlockedUserService.java
- **Implementation (serviceImpl)**: Implements service logic.
  - UserServiceImpl.java, AdServiceImpl.java, ReferralServiceImpl.java
  - WalletServiceImpl.java, CommentServiceImpl.java, FlagServiceImpl.java

---

8. Resources (resources)

This folder contains configuration files and static resources.

- application.properties – Defines application configurations.
- static/profile-pics/ – Stores user profile pictures.

---

9. Test Package (test)

This package contains test cases for the application.

- AdWatchRewardsAppApplicationTests.java – Tests the main application.

Conclusion

The **AdWatch Rewards** app follows a modular design, with clear separation of concerns, making it scalable and maintainable. The layered architecture ensures smooth interaction between controllers, services, repositories, and entities.