Overview of NewsApp Application

NewsApp is a comprehensive Android application designed to provide users with up-to-date news articles from various categories, allowing them to stay informed and engaged with the latest happenings. The app offers a user-friendly interface with features such as browsing articles, reading detailed content, leaving comments, and liking articles.

The application's core functionalities include:

1. “User Authentication and Profile Management”: Users can sign in with their credentials or register as new users to access personalized features. Each user has a profile displaying their username and profile picture.

2. “Browsing News Articles”: Users can explore a diverse range of news articles categorized into various sections, including sports, politics, technology, entertainment, and more.

3. “Reading Articles”: Users can view detailed articles by clicking on their titles, allowing them to read the entire content, view the article source, and check the publication date.

4. “Commenting on Articles”: Registered users can post comments on articles to express their opinions and engage in discussions with other users.

5. “Like Functionality”: Users can show their appreciation for articles by giving them a "like." The application tracks the number of likes each article receives.

6. “Administrator Dashboard”: An exclusive dashboard is provided for administrators to manage the app's content. Administrators can view all articles, edit existing articles, and delete unwanted articles.

7. “Real-time Updates”: The application fetches news articles from an external API, ensuring that users receive real-time updates on the latest events and news stories.

8. “Profile and Logout”: Users can access their profiles, where they can view their usernames and profile pictures. Additionally, users can log out to protect their account security.

MainActivity Documentation

Overview

The MainActivity is the entry point of the application and serves as the login screen. It allows the user to log in using their username and password or navigate to the registration screen if they are new users. The MainActivity handles user input, performs the login process, and communicates with the BackgroundWorker class to send login credentials to the server for authentication.

Class Description

“MainActivity extends AppCompatActivity”

The “MainActivity” class is a subclass of the “AppCompatActivity” class provided by the Android framework. It handles user interactions on the login screen and manages the login process.

UI Elements

The MainActivity layout (“activity_main.xml”) consists of the following UI elements:

1. “EditText UsernameEt”: An EditText widget to input the username.
2. “EditText PasswordEt”: An EditText widget to input the password.

Methods

“onCreate(Bundle savedInstanceState)”

This method is called when the activity is created. It sets up the layout and initializes the “UsernameEt” and “PasswordEt” EditText fields. The focus is automatically set on the “UsernameEt” field for a smoother user experience.

“onLogin(View view)”

This method is invoked when the user taps the "Login" button. It retrieves the username and password entered by the user, creates an instance of the “BackgroundWorker” class, and executes the login process in the background by calling the “execute()” method. The “BackgroundWorker” class is responsible for making an HTTP request to the server to authenticate the user.

“OpenReg(View view)

This method is called when the user clicks the "Register" button. It starts the Register activity by creating an explicit intent and navigating to the “Register” class.

Important Notes

1. The “onLogin” method initiates the login process by passing the entered username and password to the “BackgroundWorker” class. The BackgroundWorker class handles the HTTP request and server communication asynchronously, ensuring that the UI remains responsive during the login process.

2. The “OpenReg” method starts the Register activity, allowing new users to register and create an account.

Usage

The “MainActivity” class serves as the starting point of the application and is launched when the app is opened. It enables users to log in with their credentials or navigate to the registration page to create a new account. The login process is handled asynchronously, ensuring a smooth user experience and responsiveness. If the user enters valid credentials, the “BackgroundWorker” class communicates with the server to authenticate the user and redirect them to the appropriate activity based on their user type (admin or regular user). If the login is not successful, an appropriate message is displayed to the user.

The MainActivity class is crucial for managing user authentication and providing a secure login process for the application. It forms the foundation for user interactions and access control throughout the app.

BackgroundWorker Documentation

Overview

The “BackgroundWorker” class is responsible for handling background tasks related to user authentication and registration. It is used to perform HTTP requests to the server to login or register a user based on the provided credentials. The class extends the “AsyncTask” class, which allows it to execute background operations asynchronously while keeping the UI thread responsive.

Class Description

“BackgroundWorker extends AsyncTask<String, Void, String>“

The “BackgroundWorker” class is a subclass of the “AsyncTask” class, which provides a simple way to perform background operations and publish results on the UI thread.

Methods

“doInBackground(String... params)”

This method is executed in the background thread and handles the login or registration process, depending on the provided parameters. It constructs and sends HTTP requests to the server, retrieves the response, and returns it as a string.

“onPreExecute()”

This method is executed before the background task is started. It initializes the “AlertDialog” that will be used to display login status messages to the user.

“onPostExecute(String result)”

This method is executed after the background task is completed. It receives the result returned from the “doInBackground” method. The “onPostExecute” method checks the login or registration status and displays an appropriate message using the “AlertDialog”. If the login is successful, it also

navigates the user to the appropriate activity (AdminDashboard or ProfileActivity) based on their user type.

“onProgressUpdate(Void... values)”

This method is not used in this implementation and left empty.

Important Notes

1. The “BackgroundWorker” class plays a crucial role in handling user authentication and registration. It communicates with the server to validate user credentials and receives the server's response as a JSON string.

2. The class uses an “AlertDialog” to display login status messages, ensuring that the user is informed about the success or failure of the login process.

Usage

The “BackgroundWorker” class is used in the “MainActivity” to handle the login process and in the “Register” activity to handle the user registration process. It is initiated and executed by calling its “execute” method with appropriate parameters.

When the “BackgroundWorker” class is used for login, it sends the entered username and password to the server for validation. The server checks the credentials and responds with a JSON string containing the login status, user ID, and image URL. The “BackgroundWorker” class parses the JSON response and, if the login is successful, navigates the user to the appropriate activity based on their user type (admin or regular user).

When the “BackgroundWorker” class is used for registration, it sends the entered user details (name, email, and password) to the server to create a new account. The server processes the request and responds with a message indicating the success or failure of the registration process.

ProfileActivity Documentation

Overview

The “ProfileActivity” class is responsible for displaying the user's profile information and a list of articles authored by the user. It fetches the user's information and their authored articles from the server using HTTP requests and displays them in the activity. The user's profile information includes their username and profile image, while the list of articles includes their titles, categories, dates, and content.

Class Description

“ProfileActivity extends AppCompatActivity”

The “ProfileActivity” class is an activity that displays the user's profile and the list of articles they have authored.

Components

RecyclerView

A “RecyclerView” is used to display the list of articles authored by the user. It uses a custom adapter, “MyAdapter”, to populate the data into the list.

TextView

A “TextView” is used to display the user's username as a welcome message.

ImageView

An “ImageView” is used to display the user's profile image fetched from the server.

Button

A “Button” is used to allow the user to log out from the application.

Methods

“onCreate(Bundle savedInstanceState)”

This method is called when the activity is created. It initializes the activity and sets up the UI components. It fetches the user's ID and profile image URL from the intent extras and displays the user's username as a welcome message. It also sets up the logout button's click listener to handle user logout.

“onLogout()”

This method is called when the user clicks the logout button. It navigates the user back to the “MainActivity” and logs them out of the application.

“loadRecyclerViewData()”

This method is responsible for fetching the user's authored articles from the server using HTTP requests. It uses the Volley library to send a GET request to the server and receives a JSON response containing the article data. The method parses the JSON data and populates the “RecyclerView” with the list of articles using the “MyAdapter” custom adapter.

Important Notes

1. The “ProfileActivity” displays the user's profile information and a list of their authored articles, providing users with a personalized experience.

2. The user's profile image is fetched from the server using Glide, a library for loading and displaying images efficiently.

3. The “loadRecyclerViewData()” method fetches the article data asynchronously, ensuring a smooth user experience even when fetching large amounts of data from the server.

Usage

The “ProfileActivity” class is launched when the user logs in successfully. It receives the user's ID and profile image URL as intent extras and uses this information to fetch the user's profile data from the server and display it in the activity. The list of authored articles is fetched and displayed in the “RecyclerView”, allowing users to view the articles they have written.

The “ProfileActivity” enhances user engagement and personalization, as it provides a dedicated space for users to view their profile and authored articles. Users can also log out from the application using the logout button, which navigates them back to the login screen (“MainActivity”) and ends their session.

MyAdapter Documentation

Overview

The “MyAdapter” class is a custom adapter used to populate a “RecyclerView” with a list of “ListItem” objects. Each item in the list represents an article fetched from the database and contains information such as the article title, category, date, and source.

Class Description

“MyAdapter extends RecyclerView.Adapter<MyAdapter.ViewHolder>“

The “MyAdapter” class extends “RecyclerView.Adapter” and is responsible for inflating the layout for each item in the list and binding the data to the corresponding views.

Components

ViewHolder

A “ViewHolder” class is used to hold references to the views of each item in the list. It contains references to the article title (“title”), category (“category”), date (“date”), source (“src”), and image (“image”) views.

Constructor

“MyAdapter(Context context, List<ListItem> listItems, String userId)”

- “context”: The context of the activity or fragment using the adapter.
- “listItems”: A list of “ListItem” objects representing the articles authored by the user.
- “userId”: The ID of the user currently logged in.

Methods

“onCreateViewHolder(ViewGroup parent, int viewType)”

This method is responsible for creating a new “ViewHolder” instance for each item in the list. It inflates the layout for each item from the “list_item.xml” layout file and returns a new “ViewHolder” object.

“onBindViewHolder(ViewHolder holder, int position)”

This method is called for each item in the list to bind the data to the views in the “ViewHolder”. It gets the “ListItem” object corresponding to the current position and sets the data (title, category, date, source, and image) to the appropriate views.

“getItemCount()”

This method returns the total number of items in the list.

“ViewHolder”

The “ViewHolder” class holds references to the views of each item in the list. It sets an “OnClickListener” on the item view, allowing users to click on an article and view its details in the “postActivity”.

“onClick(View v)”

This method is called when an item in the list is clicked. It gets the position of the clicked item, retrieves the corresponding “ListItem” object, and creates an intent to launch the “postActivity”. The intent carries the necessary data (article content, title, source, date, user ID, article ID, and article image URL) to be displayed in the “postActivity”.

Important Notes

1. The “MyAdapter” class is essential for displaying the list of articles authored by the user in the “ProfileActivity”.

2. Each item in the list contains data such as the article title, category, date, source, and image, which are bound to the appropriate views in the “ViewHolder”.

3. When a user clicks on an article, the “onClick” method in the “ViewHolder” creates an intent to launch the “postActivity”, where the full details of the article are displayed.

4. The “MyAdapter” efficiently handles the list of articles and provides a smooth scrolling experience in the “RecyclerView”.

Usage

The “MyAdapter” class is used in the “ProfileActivity” to populate the “RecyclerView” with a list of articles authored by the user. When the “ProfileActivity” is launched, the list of articles is fetched from the server, and the “MyAdapter” is initialized with the retrieved data. The “RecyclerView” is then populated with the list of articles, and each item is displayed with its respective title, category, date, and image. When the user clicks on an article, the “onClick” method in the “ViewHolder” is triggered, and the full details of the article are displayed in the “postActivity”.

ListItem Documentation

Overview

The “ListItem” class represents an article authored by the user. Each “ListItem” object contains information about the article, such as its ID, title, category, date, content, source, and image URL.

Class Description

“ListItem”

The “ListItem” class is a simple POJO (Plain Old Java Object) used to encapsulate the data of an article. It stores the information about the article in private fields and provides getter methods to access this information.

Fields

“article_id”

- Type: “Integer”
- Description: The unique identifier of the article.

“title”

- Type: “String”
- Description: The title of the article.

“category”

- Type: “String”
- Description: The category of the article.

“date”

- Type: “String”
- Description: The date when the article was published.

“content”

- Type: “String”
- Description: The content or main text of the article.

“src”

- Type: “String”
- Description: The source or origin of the article.

“image”

- Type: “String”
- Description: The URL of the image associated with the article.

Constructor

“ListItem(Integer article_id, String title, String category, String date, String content, String src, String image)”

- Description: The constructor initializes a “ListItem” object with the provided data for an article.

Methods

“getArticle_id()”

- Return Type: “Integer”
- Description: Returns the unique identifier (ID) of the article.

“getTitle()”

- Return Type: “String”
- Description: Returns the title of the article.

“getCategory()”

- Return Type: “String”

- Description: Returns the category of the article.

“getDate()”

- Return Type: “String”

- Description: Returns the date when the article was published.

“getContent()”

- Return Type: “String”

- Description: Returns the content or main text of the article.

“getSrc()”

- Return Type: “String”

- Description: Returns the source or origin of the article.

“getImage()”

- Return Type: “String”

- Description: Returns the URL of the image associated with the article.

Important Notes

1. The “ListItem” class represents an article's data in a structured manner.

2. Each “ListItem” object corresponds to a single article authored by the user and contains information such as the article's title, category, date, content, source, and image URL.

3. The “ListItem” objects are used in the “MyAdapter” class to populate the “RecyclerView” in the “ProfileActivity”.

4. The “ListItem” class does not have any setter methods, making the objects immutable once created. The data can only be accessed using the provided getter methods.

Usage

The “ListItem” objects are created and populated with data when the user's articles are fetched from the server in the “ProfileActivity”. The “MyAdapter” class then uses these “ListItem” objects to display the list of articles in the “RecyclerView”. When the user clicks on an article, the “onClick” method in the “ViewHolder” retrieves the corresponding “ListItem” object and extracts its data to display in the “postActivity”.

postActivity Documentation

Overview

“postActivity” is an Android activity class responsible for displaying the details of a single article and handling user interactions related to comments and likes. The activity allows users to view the content of an article, read and add comments, and like the article.

Class Description

“postActivity”

The “postActivity” class extends “AppCompatActivity” and provides the functionality to display article details and manage comments and likes.

Fields

- “postContentTextView”: A “TextView” used to display the content of the article.
- “Headline”: A “TextView” used to display the title of the article.
- “Source”: A “TextView” used to display the source of the article.
- “Date”: A “TextView” used to display the publication date of the article.
- “Cover”: An “ImageView” used to display the cover image of the article.
- “likeBtn”: A “Button” used to handle the like action.
- “commentEditText”: An “EditText” used for entering comments.
- “submitCommentButton”: A “Button” used to submit the comment.

Methods

“onCreate(Bundle savedInstanceState)”

- Description: This method is called when the activity is created. It initializes the UI elements, retrieves article details from the “Intent”, fetches comments data from the server, and sets up the comment RecyclerView.

“addLike()”

- Description: This method is called when the user clicks on the like button. It sends a request to the server to add a like for the current user and article.

“storeLikeInDataBase()”

- Description: This method is used to store the like in the database by making a network request to the server using the “BackgroundWorkerLike” class.

“submitComment()”

- Description: This method is called when the user clicks on the submit comment button. It gets the comment text from the “commentEditText” and stores the comment in the database by making a network request to the server.

“storeCommentInDatabase(String commentText)”

- Description: This method sends a POST request to the server to store the user's comment in the database. It uses Volley to handle the network request and communicates with the server using PHP scripts.

“GetCommentsRequest”

- Description: This is an inner class that extends “AsyncTask”. It is used to fetch comments data from the server in the background and update the comment RecyclerView with the retrieved data.

“parseCommentsData(String response)”

- Description: This method parses the JSON response received from the server, extracts the comments data, and updates the comment RecyclerView with the new data.

Important Notes

1. The “postActivity” displays the content of an article and allows users to interact by adding comments and liking the article.
2. The comments and likes data are fetched from the server using HTTP requests to PHP scripts.
3. The “postActivity” class communicates with the server using the “BackgroundWorkerLike” class for handling likes and “Volley” library for fetching comments data.
4. The class follows the best practice of using background threads (like “AsyncTask”) to perform network operations to avoid blocking the main UI thread.
5. The comments and likes data are displayed using RecyclerView and Adapters to ensure smooth scrolling and efficient memory usage.

ListComment Class Documentation

Overview

The “ListComment” class is a simple model class representing a comment for a specific article. It stores the information related to a comment, such as the title of the article, the user who made the comment, the comment text, and the URL of the user's profile image.

Class Description

“ListComment”

The “ListComment” class is a model class used to represent a comment for a specific article.

Fields

- “title1”: A “String” representing the title of the article associated with the comment.
- “user”: A “String” representing the user who made the comment.
- “comment”: A “String” representing the content of the comment.
- “img_url”: A “String” representing the URL of the user's profile image.

Constructor

“ListComment(String title1, String user, String comment, String img_url)”

- Description: The constructor initializes a new “ListComment” object with the provided data.

Methods

“getTitle1()”

- Description: This method returns the title of the article associated with the comment.

“getUser()”

- Description: This method returns the username of the user who made the comment.

“getComment()”

- Description: This method returns the content of the comment.

“getImg_url()”

- Description: This method returns the URL of the user's profile image.

Important Notes

1. The “ListComment” class is a basic data container used to store comment information related to a specific article.

2. The class follows the common Java bean convention, providing getter methods for accessing the fields' values.

3. The “ListComment” objects are used to populate the comment RecyclerView in the “postActivity” class, allowing users to view and read the comments for a particular article.

MyAdapter2 Class Documentation

Overview

The “MyAdapter2” class is a custom RecyclerView adapter used to display a list of comments for a specific article. It binds the comment data to the corresponding views in the RecyclerView item layout.

Class Description

“MyAdapter2”

The “MyAdapter2” class is a custom RecyclerView adapter used to display a list of comments for a specific article.

Fields

- “context”: The “Context” object representing the context in which the adapter is being used.

- “listComments”: A “List” of “ListComment” objects representing the comments to be displayed in the RecyclerView.

Constructor

“MyAdapter2(Context context, List<ListComment> listComments)”

- Description: The constructor initializes a new “MyAdapter2” object with the provided context and list of comments.

Methods

“onCreateViewHolder(ViewGroup parent, int viewType)”

- Description: This method inflates the layout for a single item of the RecyclerView.

“onBindViewHolder(ViewHolder holder, int position)”

- Description: This method binds the comment data to the corresponding views in the RecyclerView item layout.

“getItemCount()”

- Description: This method returns the number of comments in the list.

Inner Class

“ViewHolder”

The “ViewHolder” class is a nested inner class representing a single item view of the RecyclerView. It holds references to the views inside the item layout.

Fields

- “user”: A “TextView” representing the username of the user who made the comment.
- “comment”: A “TextView” representing the content of the comment.
- “image”: An “ImageView” representing the user's profile image.

Constructor

"ViewHolder(View itemView)"

- Description: The constructor initializes a new "ViewHolder" object with the provided itemView.

Important Notes

1. The "MyAdapter2" class is used to populate the RecyclerView in the "postActivity" class with comments related to a specific article.

2. It uses the "Glide" library to load and display the user's profile image associated with each comment.

3. The "ViewHolder" class holds references to the views in the "list_comment" layout, which are then updated with the comment data during the binding process.

4. The "MyAdapter2" class works in conjunction with the "ListComment" class, which provides the comment data to be displayed in the RecyclerView.

AdminDashboard Class Documentation

Overview

The "AdminDashboard" class represents the activity where the admin user can view and manage articles. It displays a list of articles in a RecyclerView and allows the admin to delete articles from the list.

Class Description

"AdminDashboard"

The "AdminDashboard" class is an AppCompatActivity that serves as the dashboard for the admin user.

Fields

- “URL_DATA”: A constant “String” representing the URL to fetch article data from the server.

- “recyclerView”: A “RecyclerView” to display the list of articles.

- “adapter”: An instance of “MyAdapter3” used as the RecyclerView adapter.

- “listItems”: A “List” of “ListItem” objects representing the articles to be displayed in the RecyclerView.

- “article_id”: An “Integer” representing the ID of the selected article to be deleted.

- “userUrl”: A “String” representing the URL to fetch user data from the server.

- “progressDialog”: A “ProgressDialog” to show while loading data from the server.

- “username_view”: A “TextView” to display the admin's username.

- “logoutBtn”: A “Button” to handle the logout action.

Methods

“onCreate(Bundle savedInstanceState)”

- Description: This method is called when the activity is created. It initializes the views, sets up the RecyclerView, and loads the article data from the server.

“onLogout()”

- Description: This method handles the logout action. It navigates the admin user back to the main login screen.

“loadRecyclerViewData()”

- Description: This method fetches the article data from the server and updates the RecyclerView with the fetched data.

“onDeleteClick(int position)”

- Description: This method is called when the admin clicks the delete button for a specific article in the RecyclerView. It sends a request to the server to delete the selected article and updates the RecyclerView accordingly.

Important Notes

1. The “AdminDashboard” class is designed for admin users to manage articles in the app.

2. It uses the “MyAdapter3” class to display the list of articles in the RecyclerView.

3. The “MyAdapter3” class is assumed to have a method called “OnDeleteClickListener” to handle the delete button clicks in the RecyclerView items.

4. When the admin clicks the delete button for an article, the “onDeleteClick” method is triggered, and the corresponding article is deleted from the server and removed from the RecyclerView.

MyAdapter3 Class Documentation

Overview

The “MyAdapter3” class is a RecyclerView adapter used in the “AdminDashboard” activity to display a list of articles for the admin user. It also handles click events on the RecyclerView items, including deleting an article from the list.

Class Description

“MyAdapter3”

The “MyAdapter3” class is a RecyclerView adapter that displays articles in the “AdminDashboard” activity.

Fields

- “context”: A “Context” object to access the application's resources and services.

- “listItems”: A “List” of “ListItem” objects representing the articles to be displayed in the RecyclerView.

- “onDeleteClickListener”: An instance of the “OnDeleteClickListener” interface to handle delete button clicks for articles.

Methods

“MyAdapter3(Context context, List<ListItem> listItems, OnDeleteClickListener listener)”

- Description: Constructor for “MyAdapter3” class.
- Parameters:
  - “context”: The context of the application/activity.
  - “listItems”: The list of articles to be displayed.
  - “listener”: An instance of the “OnDeleteClickListener” interface.

“onCreateViewHolder(ViewGroup parent, int viewType)”

- Description: This method is called when the RecyclerView needs a new “ViewHolder” for an item view.
- Parameters:
  - “parent”: The “ViewGroup” into which the new “View” will be added after it is bound to an adapter position.
  - “viewType”: The view type of the new “View”.
- Returns: A new “ViewHolder” that holds an item view.

“onBindViewHolder(ViewHolder holder, int position)”

- Description: This method is called to display the data at a specified position in the RecyclerView.
- Parameters:
  - “holder”: The “ViewHolder” that should be updated to represent the contents of the item at the given position in the data set.
  - “position”: The position of the item within the adapter's data set.

“getItemCount()”

- Description: This method returns the total number of items in the data set held by the adapter.

- Returns: The total number of items in the RecyclerView.

“OnDeleteClickListener”

- Description: An interface to handle delete button clicks for articles in the RecyclerView.

“ViewHolder”

- Description: An inner class that represents the view holder for each item in the RecyclerView.

- Fields:

  - “title”: A “TextView” to display the title of the article.

  - “category”: A “TextView” to display the category of the article.

  - “date”: A “TextView” to display the date of the article.

  - “src”: A “TextView” to display the source of the article.

  - “image”: An “ImageView” to display the image of the article.

  - “btnDelete”: A “Button” to handle the delete action for the article.

- Methods:

  - “ViewHolder(View itemView)”: Constructor for the “ViewHolder” class. Initializes the views and sets up click listeners for the item view and delete button.

Important Notes

1. The “MyAdapter3” class is used to display a list of articles for the admin user in the “AdminDashboard” activity.

2. It also handles click events on the RecyclerView items, allowing the admin to delete articles from the list.

3. When the delete button is clicked for an article, the “onDeleteClick” method of the “OnDeleteClickListener” interface is triggered, and the “AdminDashboard” activity handles the delete action for the selected article.