

# ~ Social Scheduler ~

İbrahim Karaca  
Ahmet Deniz Gelir

## Detailed Design Report

( v1 )

19 May 2023

## 1. Introduction

The Android app called "Social Scheduler" combines scheduling features with social media elements, allowing users to plan and share their weekly activities. By receiving likes and feedback, routines can be sorted based on popularity.

## 2. Details

### 2.1 Java Classes

The following section provides a concise overview of the classes that will be utilized in our Android application's UI design, constituting the foundation of XML technology. Each class encompasses various variables and methods, and certain classes exhibit interconnectedness with one another. These relationships are further explained in a UML diagram, offering a more comprehensive description.

#### 2.1.1 *AddCommentActivity*

This is a class for the activity that lets users add the comments of a particular shared schedule(Post).

- User user // The user retrieved from firebase.
- Button commentBtn // To add a comment.
- EditText contentInput // The string of comments that user wants to add.

#### 2.1.2 *BuildShareFragment*

The class that allows users to create events and share with others.

- User user // The user retrieved from firebase.
- View rootView // The rootView of fragment to access other views.
- int day // Represents the selected day
- Schedule schedule // The schedule that user is working on
- void displaySchedule() // displays the schedule on the screen
- void setDay(int day) // sets the day according to the user input
- String dayString(int day) // converts day index to day string.

### **2.1.3 ColorUtil**

This is a class for creating colors of events.

- static int strToColor(String str) // Converts the string formed color to hexcode.

### **2.1.4 Comment**

This is a class representing comments that are stored in firebase.

- String comment // string form of comment
- User user
- String getComment()
- User getUser()

### **2.1.5 CommentActivity**

This is a class for the activity that lets users view the other users comments.

- Post post // the post user that is seen on the screen
- void displaySchedule() // displays the schedule on the screen.
- int findAvatar(int avatarIndex) // converts the avatar index to avatar hexcode
- void displayUserCard() // displays the user information
- void displayComments() // displays the comments

### **2.1.6 CommentDB**

This is a class for the data transition between comments and firebase.

- String uid // user id
- String username
- void setUsername(String username)
- String getUid()

### **2.1.7 CreateEventActivity**

This is a class for the activity that lets users create events.

- User user // The user retrieved from firebase.
- Button addEvent //

### **2.1.8 DeleteDialog**

This is a class for the activity that lets users delete the clicked event on build & share fragment.

- ArrayList<Event> events
- int index // the index of selected event
- BuildShareFragment fragment // to re-display after deleting the selected event.

### **2.1.9 DetailsActivity**

This is a class for the activity that lets users add the comments of a particular shared schedule(Post).

- ArrayList<Event> events
- RelativeLayout[] days // The represented days on the UI design.
- TextView likecount, dislikecount
- User user
- Post post

- void displayUserCard() // displays the user information
- void displaySchedule() //displays the schedule according to the time blocks.
- int dpToInt() // converts the pixel depth coming from xml to integer value.
- void createDays() // creates the days array.

#### **2.1.10 Like**

This class prevents users from adding more than one like to the same post.

- boolean filler // if users already added a like, this turns to be false

#### **2.1.11 Event**

This is a class representing the events on the time blocks.

- int start // start time
- int end // end time
- String name // event name
- int color // event color

#### **2.1.12 ExploreFragment**

This is a class for the activity that lets users search other users' posts.

- ArrayList<Post> posts // other users' posts
- GridLayout grid // the grid layout to add posts
- View rootView // to access the other view on screen
- ImageView searchIcon // to search a specific post
- void search() // displays the searched posts
- boolean doSearchMatch(String searched, String title) // returns true if the searched post is found
- void sortPosts(ArrayList<Post> posts) // sorts the posts in accordance with like rates.
- void displayCards() // displays the posts

#### **2.1.13 ImportActivity**

This is a class for the activity that lets users import other users' schedules into theirs.

- RelativeLayout[] days
- Schedule primarySchedule, secondarySchedule, currentPermutation // represents the schedules in case a conflict occurs.
- void remerge() // resolves a conflict
- void createDays() // creates the days array
- void displaySchedule() // displays the schedule

#### **2.1.14 Dislike**

This class prevents users from adding more than one dislike

- boolean filler // if users already added a like, this turns to be false

#### **2.1.15 LoggedIn**

This is a class for the activity that allows users to log in to their accounts.

- FirebaseAuth auth // authorization information
- FirebaseFirestore db // database retrieved from cloud firestore
- String userSid // user id

#### **2.1.16 MainActivity**

This is a class that represents the initial page of our application.

- FirebaseAuth auth // authorization information
- FirebaseFirestore db // database retrieved from cloud firestore
- EditText passField // password input
- EditText emailField // email input
- Button SignUpBtn // signing up activity button

### **2.1.17 Post**

This is a class that represents the posts shared in the explore page.

- int like
- int dislike
- User user
- Schedule schedule
- String title // post title
- String description // post description
- ArrayList<Comment> comments // post comments

### **2.1.18 PostDB**

This is a class for the activity that allows the data transition between firebase and posts.

- String username
- int avatarIndex
- String title // post title
- String uid // user id
- int like
- int dislike

### **2.1.19 ProfileActivity**

This is a class for the activity that lets users create their account.

- FirebaseAuth auth // authorization information
- FirebaseFirestore db // database retrieved from cloud firestore
- EditText passField // password input
- EditText emailField // email input
- Button saveBtn

### **2.1.20 Schedule**

This is a class that represents schedules created by other users.

- ArrayList<Event> events
- void addEvent(Event e) // add an event to the events ArrayList

### **2.1.21 ScheduleDB**

This is a class for the activity that allows data transition between firebase and schedules.

- Map<String, Object> eventList // a hashmap that stores events and users.
- Schedule schedule

### **2.1.22 ScheduleFragment**

This is a class for the activity that lets users check their own schedule.

- ArrayList<Event> events

- void addEvent(Event e) // add an event to the events ArrayList
- ScrollView outer // the outer Scroll Layout
- RelativeLayout[] days // the weekdays
- Random rnd // to create random colors for events

### **2.1.23 ShareActivity**

This is a class for the activity that lets users share their posts.

- Button shareBtn // To share a post
- ArrayList<Event> events

### **2.1.24 SignUpActivity**

This is a class for the activity that lets users sign up to their accounts.

- FirebaseAuth auth // authorization information
- FirebaseFirestore db // database retrieved from cloud firestore
- EditText passField // password input
- EditText emailField // email input
- Button singUpBtn
- User user

### **2.1.25 SortPostsUtil**

This is a class that implements Comparator<Post> interface in order to sort posts in accordance with their like counts.

- int compare(Post p1, Post p2) // compares the two posts according to the like counts.

### **2.1.26 User**

This is a class that represents users stored in firebase

- String username
- String name
- String bio // biography
- int avatarIndex
- boolean like100 // if a user's post achieved more than 100 likes, this becomes true.
- boolean verified // represents the verified badge
- ImageView getAvatar(View rootview) // returns avatar image based on avatar index.

### **2.1.27 UserProfileActivity**

This is a class to create users profiles according to their user data.

- void onCreate() // creates the user profile


### **2.1.28 ActivityUtil (Interface)**

- int findAvatar(int avatarIndex) // to help classes find the avatar hexcode.

## **2.2 In-App Screenshots**

### **2.2.1 Login and Sign Up Pages**

7:09



## Social Scheduler

email


password

LOGIN

Don't have an account?

CREATE ACCOUNT

7:10



## Social Scheduler

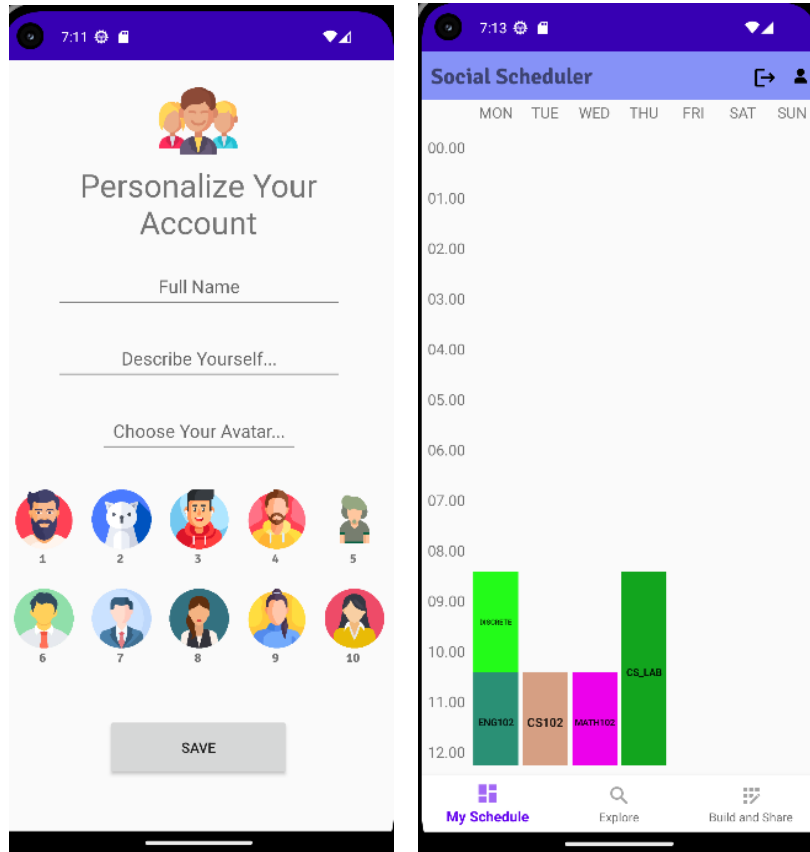
email

username

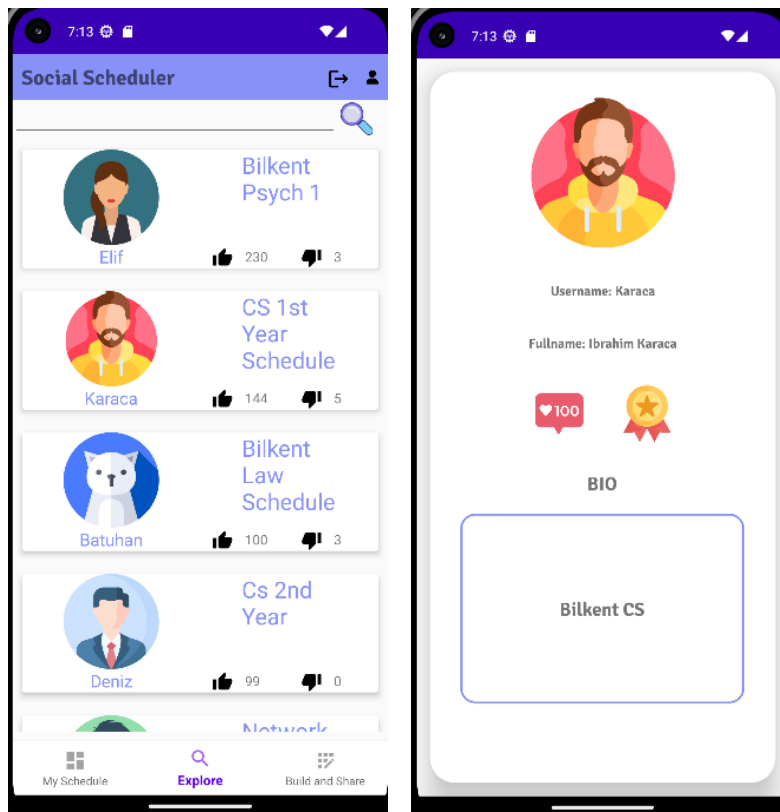
password

SIGN UP

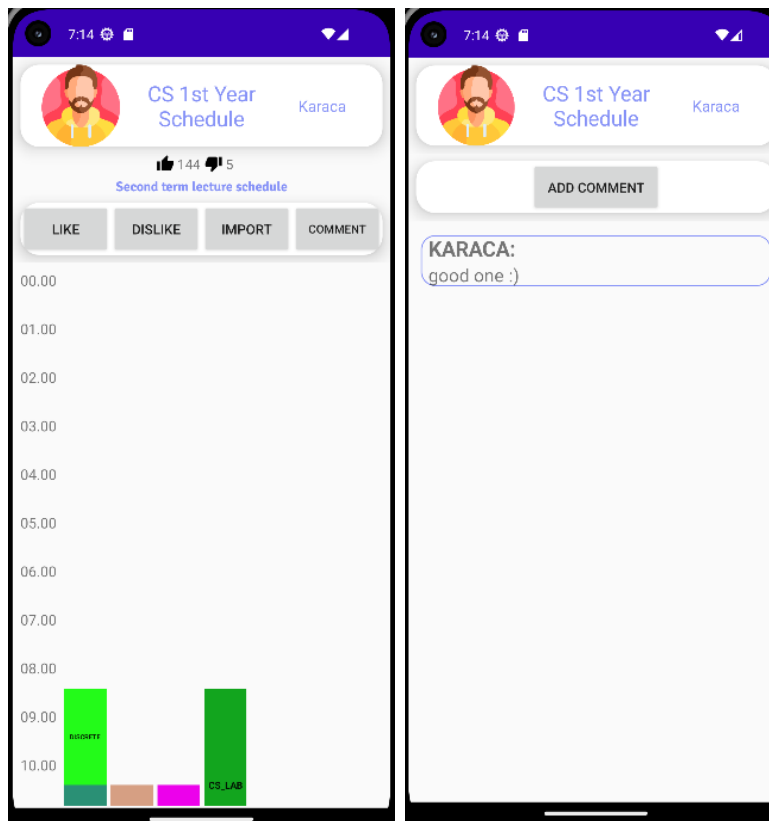
### ***2.2.2 Personalize Account and My Schedule***



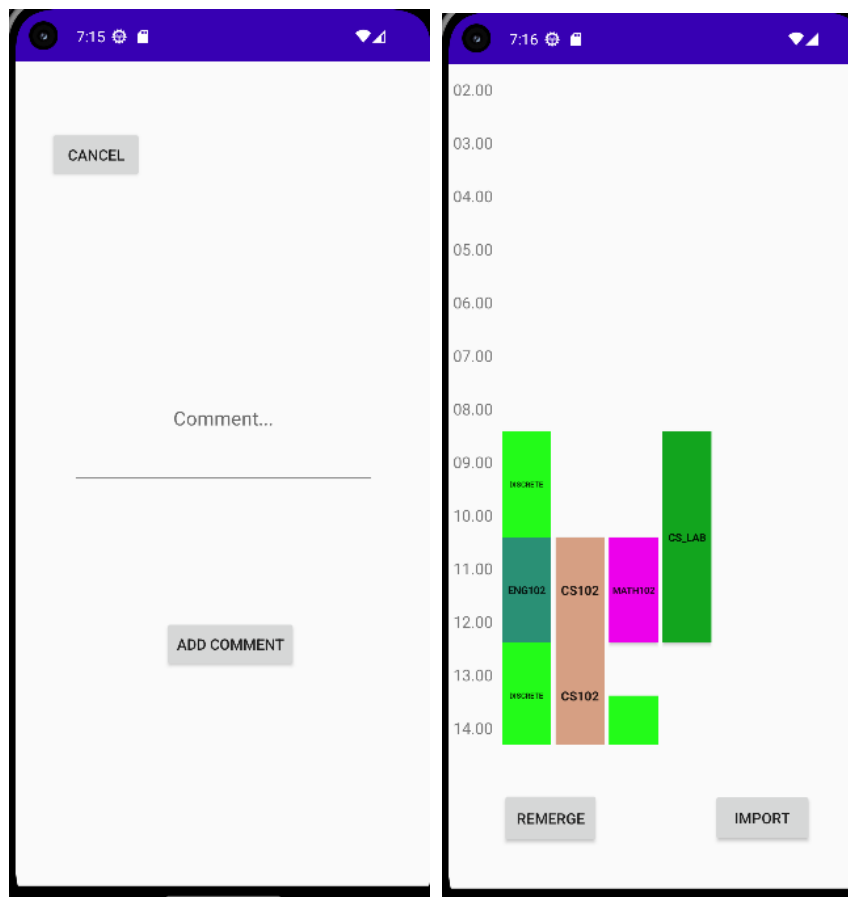
### 2.2.3 Explore and User Profiles



### 2.2.4 Detailed Post and Comments Page

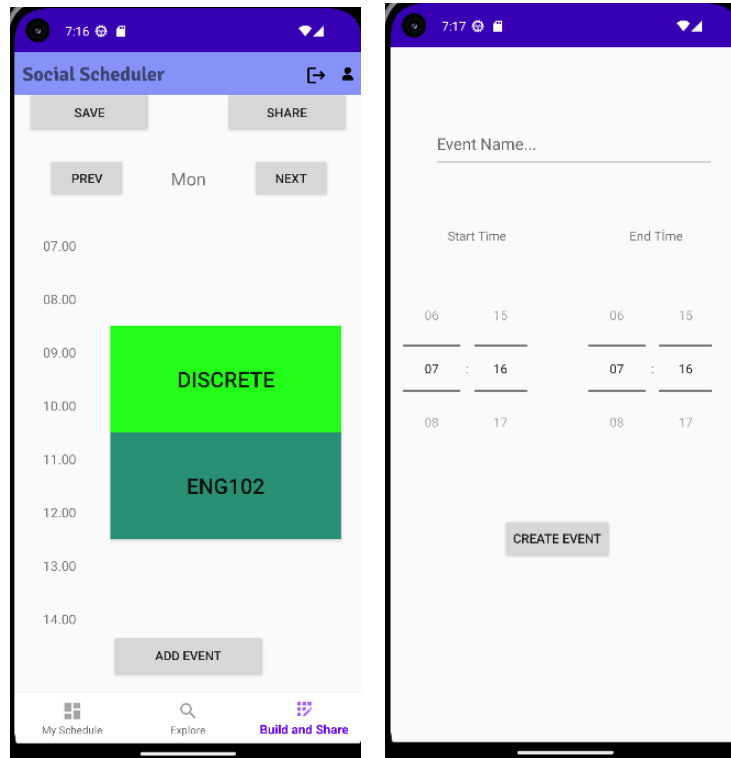


### 2.2.5 Add Comment and Import Schedule Page

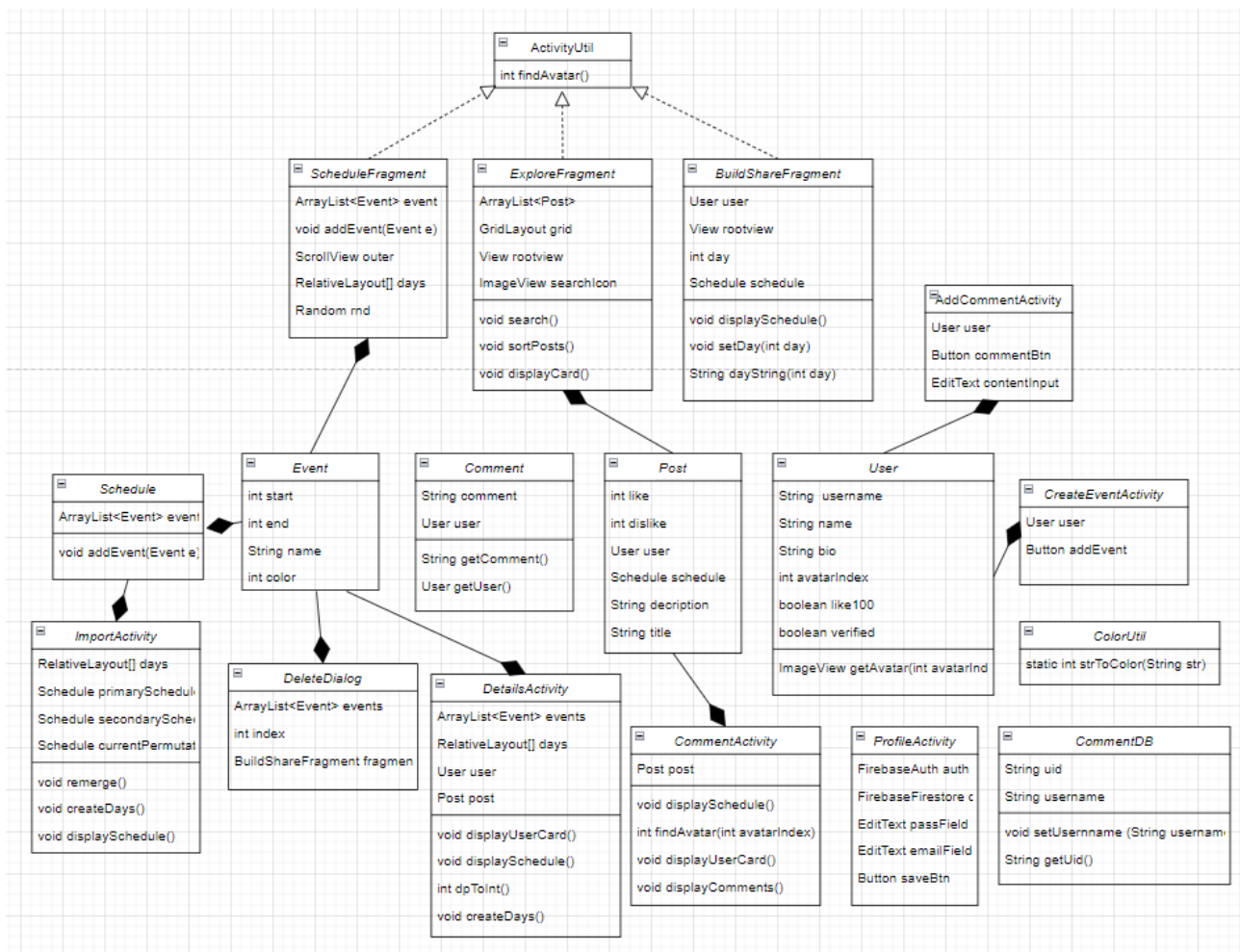


### 2.2.6 Build and Share, Add Event Pages





## 2.3 UML Diagram



### 3. Task Assignments

#### 1. İbrahim Karaca:

İbrahim has worked on the schedule algorithms and designing the schedule view. The schedule activity is an essential component that greatly impacts other sections like “My Schedule” and “Explore”. He also worked on viewing user profiles, exploring other users’ posts and creating user accounts. The visualization algorithm effectively arranges the events within the appropriate time blocks and days, including the XML layouts and UI design.

**The classes he worked on:** CommentActivity, BuildShareFragment, DetailsActivity, DeleteDialog, ExploreFragment, Post, ProfileActivity, Schedule, ScheduleFragment, SignUpActivity, SortPostsUtil, User, UserProfileActivity.

#### 2. Ahmet Deniz Gelir:

Ahmet has worked on aspects related to authentication and database access. He also developed the schedule merging algorithm, the merging UI, and the Explore section. He also worked on like-dislike functionality.

**The classes he worked on:** BuiltShareFragment, ColorUtil, CommentActivity, CommentDB, CreateEventActivity, Dislike, Event, ImportActivity, Like, LoggedIn, MainActivity, PostDB, ShareActivity, SignUpActivity, UserProfileActivity

### 4. Final Thoughts

**4.1. What we could and could not achieve?** Given our lack of experience and knowledge when starting this project, we had the opportunity to achieve a more improved user interface. However, the challenges we faced while simultaneously learning and collaborating as a group on this project made the process quite difficult.

**4.2. What did we dislike about project work?** The usage of the Android Studio IDE posed significant difficulties for us, as we constantly encountered issues that required extensive time and effort to resolve. Consequently, we developed a strong dislike towards it.

**4.3. What would we do differently if we had to start over again?** As we worked on various components of the project as a team, we encountered challenges when it came to integrating these different parts together. Over time, it became increasingly difficult to merge them seamlessly. To mitigate this issue, we decided to increase the frequency of our meetings in order to proactively address and prevent any collisions or conflicts between the different parts of the project.

**4.4. How much time did we spend on the project?** During a span of approximately two months, a significant portion of our time was dedicated to acquiring knowledge in Android Studio, XML, Java technologies, and frontend development. The learning process was quite demanding, but once we had successfully grasped these concepts, the implementation phase proved to be comparatively less challenging. As a result, we believe that the upcoming project we undertake will be easier, as we have gained valuable experience throughout this semester.

**4.5. Are we proud of what we have achieved?** Despite initially having limited knowledge about Android Studio, we firmly believe that we have successfully completed the best project we were capable of achieving.

### 5. Conclusion

To summarize, Social Scheduler is an advantageous Android application catering to users seeking efficient schedule management with added social functionalities. Users can effortlessly create their own schedules and import routines from others, free from concerns about overlapping time slots. Moreover, they can rate and comment on other users' routines, enabling the application to organize schedules based on popularity. With its user-friendly interface and intuitive design, Social Scheduler offers a convenient solution for individuals who wish to effectively manage their schedules and gain inspiration from fellow users' routines.