TED UNIVERSITY

2024 Spring Semester

CMPE 492 Senior Project II

LOW-LEVEL DESIGN REPORT

FocuZone

Team Members:

Şevval Kaplan - 37463197404

Hüseyin Sina Ceylan - 10003284536

Şevval Yardımcı - 55780306880

Eren Sakarya - 12521841116

Table of Contents

1. Introduction *3*

1.1. Object Design Trade-Offs **3**

1.1.1. Usability vs. Functionality 3

1.1.2. Reliability vs. Compatibility 4

1.1.3. Scalability vs Performance 4

1.1.4. Efficiency vs. Maintainability 4

1.1.5. Efficiency vs. Accuracy 4

1.2. Interface Documentation Guidelines **5**

1.3. Engineering standards **6**

1.4. Definitions, acronyms, and abbreviations **6**

2. Packages *7*

2.1 GUI Packages **7**

2.2 Back-End Packages **8**

3. Class Interfaces *10*

3.1. User Interface Classes **10**

3.2. Back-End Classes **17**

4. Glossary *21*

5. References *22*

1. Introduction

In the world of technology people with ADHD may have problems focusing on studying as it is a side effect of ADHD. In general, everyone might need some type of device or planner that will help them in their studies. These reasons will need a solution that will help users with ADHD stay focused and help them or others without ADHD plan better study periods.

To solve these problems and improve the study periods of users significantly, we propose the development of an application designed to aid users with ADHD reduce distractions and focus better on their studies. The main purpose of the project is scheduling works and study periods and creating a better area for work by various features such as analysing the study environment with a sound level meter and a phone camera for distracting objects.

The application will become a part of the users' daily lives as long as they are studying. The distraction free safe space that the application is going to create will be essential and against ADHD. With the software solutions provided in the application, students will be able to create better study periods and plan accordingly to their needs.

The application has several types of planners or lists to help users in need. Since in the modern world nearly everybody carries their mobile devices with them the application will be available most of the times to the user. The system will be using image processing, machine learning and visualization to process and represent the distracting objects to the user. The user then can remove or change the location of the object(s) in their study environment. The system will also be providing a sound level meter so that will help users realise they are not in an environment fit for studying if the level is too high.

1.1. Object Design Trade-Offs

1.1.1. Usability vs. Functionality

Since the FocuZone application is an application for individuals with ADHD, it was designed with great emphasis on usability. It aims to allow the user to easily perform the necessary operations within the application, benefit from our features, and complete their study session without losing focus. Our purpose is making an environment that

user will not be distracted, so our application FocuZone designed with considering usability while maintaining basic functionality and without making any sacrifices that will reduce the user's experience.

1.1.2. Reliability vs. Compatibility

FocuZone ensures reliability and user safety of the application by ensuring consistent operation of the application for users with ADHD. While making FocuZone emphasis has been placed on ensuring that the app works correctly on all Android devices without compromising security.

1.1.3. Scalability vs Performance

FocuZone is an application that focuses on optimal performance to provide users with a seamless experience, but on the other hand, scalability is considered as potential growth and this is also given importance in the project, but since performance effectively meets the needs of individuals with ADHD it has been prioritized in the construction of the project.

1.1.4. Efficiency vs. Maintainability

FocuZone focuses on the efficiency of task management and study support for ADHD patients and aims to has a seamless user experience. The focus is on providing a reliable and efficient solution for users, while not ignoring the ease of maintenance of the system.

1.1.5. Efficiency vs. Accuracy

FocuZone focuses on efficiency in the analysis of study environments with features such as Sound Level Meter and Study Environment Photo Analyzer. While accuracy is a key, the focus is on giving users quick and practical insights to improve their concentration and productivity, because of that reason efficient algorithms are prioritized.

1.2. Interface Documentation Guidelines

Our project has many classes due to its many features, and in the Class Interfaces section of our report, we will show the names of these classes, their attributes, and methods in general in the table. The structure of this table will be similar to those used in charts.

The name of the class will be at the top, and the general purpose of this class will be explained below.

Afterward, there will be a section containing the attributes used in the relevant class.

Finally, the names of the important methods I use in this class will be found in the last section of the table.

Class Name
General Purpose and Explanation of the Class
Attributes
Attributes Type and Attribute Name
Methods
Method's Name and Parameter

1.3. Engineering standards

This document for the FocuZone project adheres to Unified Modeling Language (UML) guidelines and IEEE standards, ensuring precision and accuracy in the Low-Level Design Report. UML guidelines provide a structured approach for class interfaces, diagrams, and subsystem compositions, ensuring visual consistency and traceability. Simultaneously, following the IEEE standards and writing the report according to IEEE standards guarantees the professionality of the report. This adherence to IEEE standards and UML guidelines increases the integrity of our report.

1.4. Definitions, acronyms, and abbreviations

API: Application Programming Interface.

UI: User Interface.

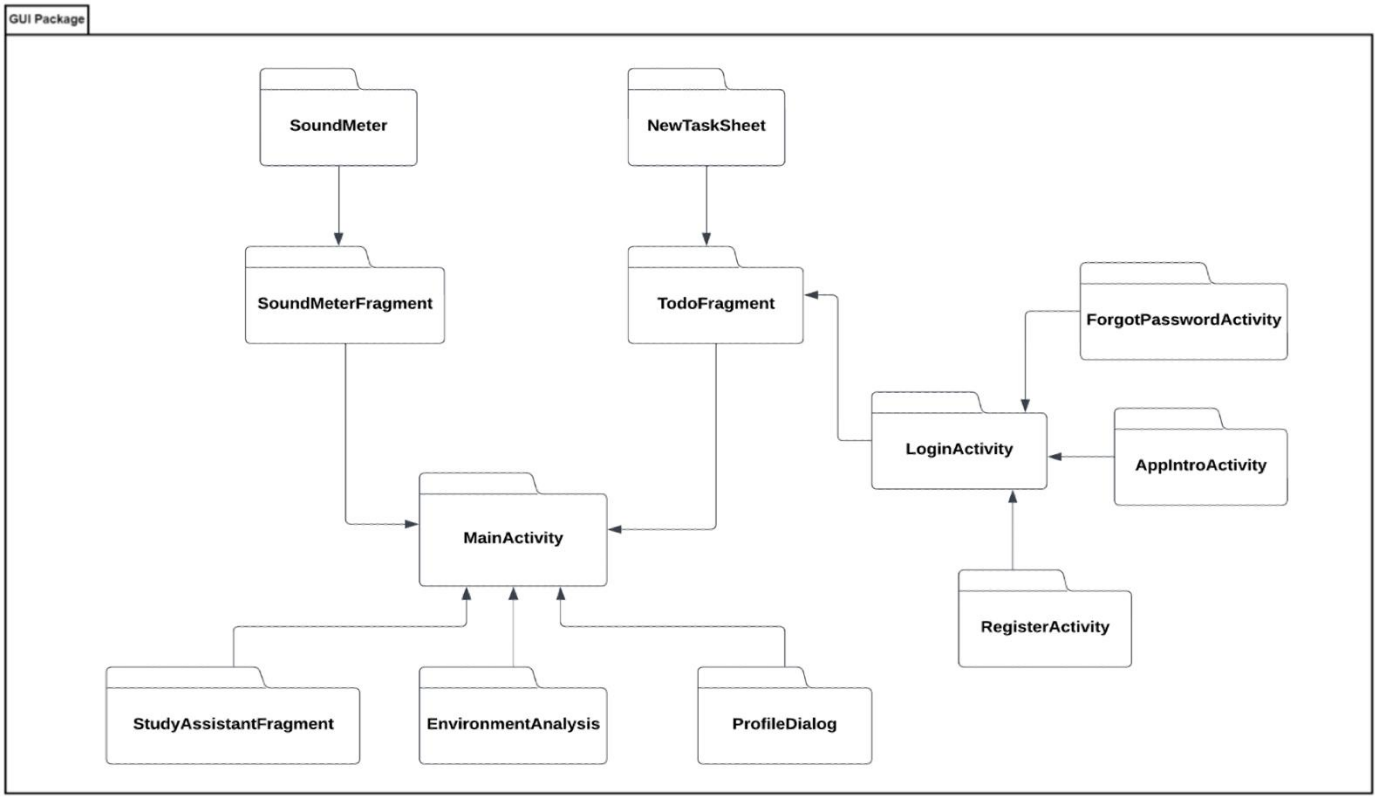
UX: User Experience.

UML: Unified Modeling Language.

IEEE: Institute of Electrical and Electronics Engineers.

2. Packages

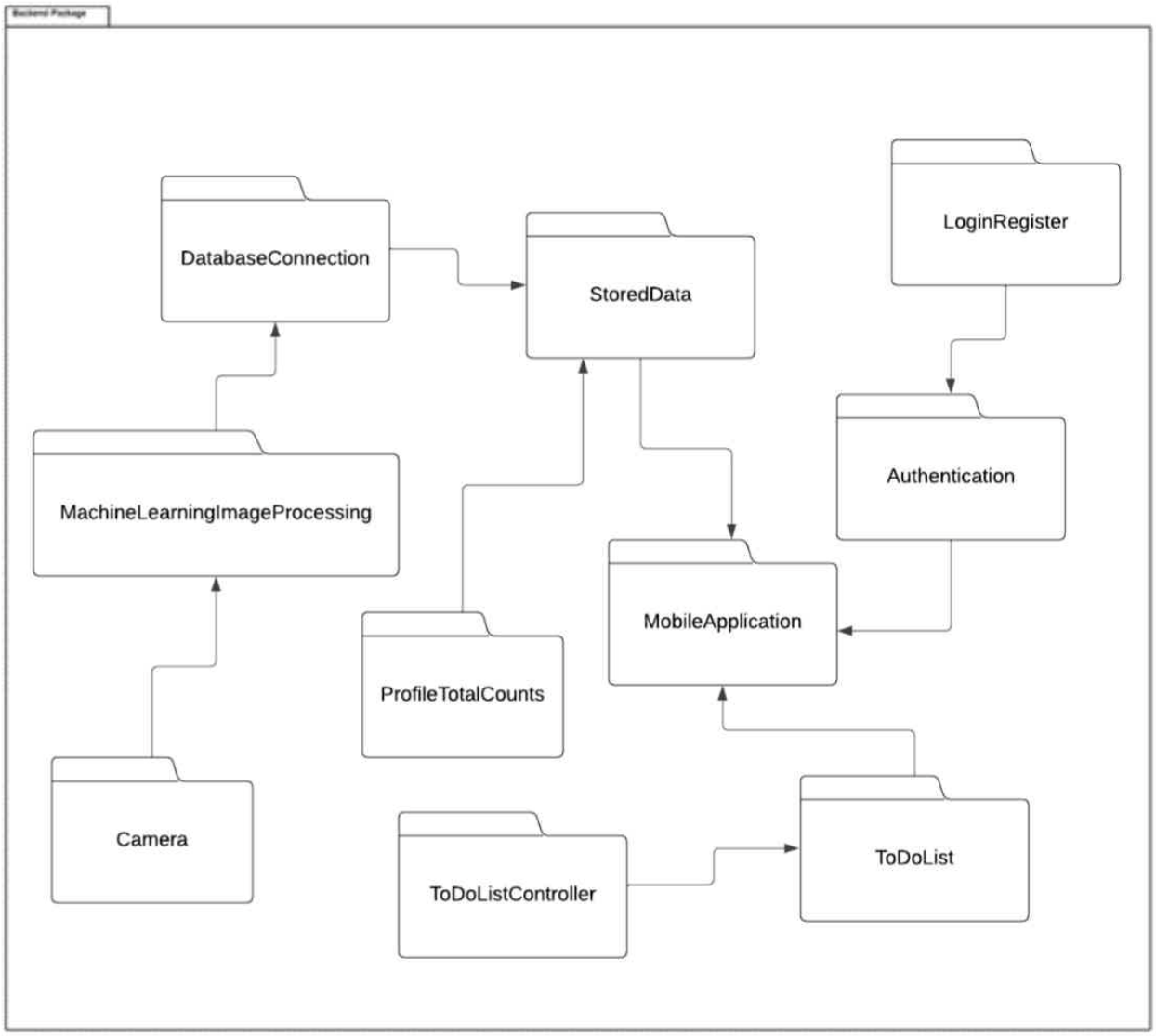
2.1 GUI Packages



In GUI Package, we see that when the users open our mobile application, the App Introduction screen welcomes our users. This page gives a brief information about our application to our users. After this step, the login screen shows up. Users who already have an account can log in to the application, while those without an account can proceed to register. If they choose to register, the registration screen will open. In case of forgotten passwords, they can access the "Forgot Password" screen by clicking the corresponding link. Upon successful login, the user is directed straight to the To-do page. Following this, the user can navigate to the relevant pages by selecting one of the icons at the bottom of the screen. These include To-Do, Pomodoro, Sound Check, and Zone Check, respectively. Similarly, users can access their profile information by pressing the button at the top right of the page. From here, users can view their task, hour, and session counts. Additionally, they can log out of the application by pressing the button with the door icon. When the user enters the To-Do page, they can view their existing tasks. Additionally, if they want to add a new task, they can press the plus button on the page. This will open the "New Task" sheet.

After entering the necessary information, they can save the new task by pressing the "Save Task" button. When they finish the task, they can press to the circle which is next to the task. Additionally, they can delete the task by pressing the delete button on the task edit screen. When the user enters to the Pomodoro page, they can create new study cycles by adjusting the study and break durations. They can stop and start these pomodoros by using the button on the middle. When user enters to the Sound Check page, they can press to the microphone icon to measure the sound level of the environment. When the user enters to the Zone Check page, they can upload the image of the environment they want to analyze.

2.2 Back-End Packages



In the Back End Package, we have several branches that all collectively connect to the MobileApplication class. The camera class is the general starting point of the image processing part. The camera class gets the necessary picture to be analyzed later on. Then the MachineLearningImageProcessing class analyses the picture for distracting, shiny objects and general room colors. The results are than stored in the StoredData class in a database via the DatabaseConnection class. These two classes act as a bridge and a storage unit for pictures and ProfileTotalCounts class. The profileTotalCounts class keeps the information of the total amount of activities a user has and displays it with some functions. Then the StoredData class connects to the MobileApplication class so that users can access them. The ToDoListController class has some functions that executes the actions needed in the ToDoList class. These actions contain add, update, and delete operations. Lastly, we have Login/Register class which will be executed while login in or registering to the app for the first time. A userID will be generated once when a user registers and an authentication token will be generated once when the user successfully logins. If the user cannot login or register properly a message will be displayed to warn them that their credentials are wrong. The generated authentication token will be used in the Authentication class to validate user information and let them use the MobileApplication class.

3. Class Interfaces

3.1. User Interface Classes

AppIntroActivity
This activity class has the feature of controlling the structure consisting of 4 intro fragments where users who log in to our application for the first time learn information about the application.
Attributes
private MyPagerAdapter adapter; private ActivityIntroBinding binding;
Methods
public void onDonePressed(View view) public void finishIntro() createFragment(int position) getItemCount() public void onNextPressed(View view) protected void onCreate(Bundle savedInstanceState)

NotificationReceiver
This class is derived from BroadcastReceiver and its purpose is to ensure that the necessary notifications are transmitted along with the channels created for notifications.
Attributes
val channelId : String val title: String val desc: String val contentIntent : Intent val contentPendingIntent: PendingIntent val notification : Notification val manager: NotificationManager val notificationId: Int
Methods
override fun onReceive(context: Context, intent: Intent?)

ProfileDialog
This class creates the profile dialog that shows how many sessions the user has completed in total, how many tasks she has completed, and how many hours she has worked in total, and also allows her to log out of the application.
Attributes
private DialogLayoutBinding binding;
Methods
protected void onCreate(Bundle savedInstanceState) private void clearSessionData() private void signOutFirebase() private void navigateToLogin() private void getInformations()

SoundMeter
This class contains some record methods that will be used in SoundMeterFragment.
Attributes
private MediaRecorder mRecorder private Context context;
Methods
public void start() public void stop() public double getAmplitude()

SoundMeterFragment

All methods that allow the user to measure the sound level of the environment are used here, and the structures that enable the result to be displayed to the user are also available here.

Attributes

private FragmentSoundMeterBinding binding;
private SoundMeter soundMeter;
private boolean isRecording;
private boolean measurementInProgress;
private boolean permissionToRecordAccepted;
private Handler handler
private static final int REQUEST_RECORD_AUDIO_PERMISSION
private String[] permissions
private double totalAmplitude
private int sampleCount
private static final int SAMPLES_TO_AVERAGE

Methods

private void startRecording()
private void stopRecording()
private float amplitudeToDb(double amplitude)
private float scaleDbToProgress(double dbValue)
public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState)
public void onResume()
public void onPause()
public void onDestroyView()
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults)

ForgotPasswordActivity

When the password is forgotten or wanted to be changed, this can be achieved thanks to this class.

Attributes

private ActivityForgotPasswordBinding binding;
private FirebaseAuth mAuth;

Methods

protected void onCreate(Bundle savedInstanceState)
private void sendPasswordResetEmail(String email)

LoginActivity
It is our class that controls and allows the user to log in to their account.
Attributes
private ActivityLoginBinding binding; private FirebaseAuth mAuth;
Methods
private void navigateToMainActivity() protected void onCreate(Bundle savedInstanceState)

RegisterActivity
Becoming a member of a new user and sending a verification to their e-mail is done thanks to this class.
Attributes
private ActivityRegisterBinding binding; private FirebaseAuth mAuth;
Methods
protected void onCreate(Bundle savedInstanceState) private void registerUser(final String email, String password) private void sendVerificationEmail()

StudyAssistantFragment

It is the class that allows us to set up working and break times for the pomodoro feature in the application and provides the requirements of the feature such as countdown, period counting, and sending notifications.

Attributes

private FragmentStudyAssistantBinding binding;
private CountDownTimer countDownTimer;
private int currentPeriod = 0;
private boolean isBreakTime = false;
private long studyTimeMillis, breakTimeMillis, timeLeftInMillis;

Methods

public View onCreateView(@NonNull LayoutInflater ViewGroup container, Bundle savedInstanceState)
private void startTimer()
private void pauseTimer()
private void setTimesFromSpinners()
private void startCountdown(long timeMillis)
private void resetTimer()
private void updatePeriodIcons()
private void resetPeriodIcons()
private void updateProgressBar()
private String formatTime(long millis)
public void onDestroyView()

NewTaskSheet

This class, which enables the creation of tasks in TodoFragment, also provides task adjustment and notification settings for todo.

Attributes

private lateinit var binding: FragmentNewTaskSheetBinding
private lateinit var taskViewModel: TaskViewModel
private var dueTime: LocalTime?
private val editable: Editable
private const val REQUEST_CODE_EXACT_ALARM_PERMISSION: int

Methods
public View onCreateView(@NonNull LayoutInflater ViewGroup container, Bundle savedInstanceState) private void startTimer() private void pauseTimer() private void setTimesFromSpinners() private void startCountdown(long timeMillis) private void resetTimer() private void updatePeriodIcons() private void resetPeriodIcons() private void updateProgressBar() private String formatTime(long millis) public void onDestroyView()

TodoFragment
It is the class related to the todo feature, where tasks are created and tracked by the user to create a daily plan.
Attributes
private FragmentTodoBinding binding; private TaskViewModel taskViewModel;
Methods
public View onCreateView(@NonNull LayoutInflater, ViewGroup container, Bundle savedInstanceState) private void setupRecyclerView() public void editTaskItem(TaskItem taskItem) public void completeTaskItem(TaskItem taskItem) public void onDestroyView() private void updateCompletionStatus(Integer completedTasks, Integer totalTasks)

EnvironmentAnalysis
This is the class where we select a photo from our gallery, send it to our system and the analysis of the photo comes back to us.
Attributes
private FragmentTodoBinding binding;
Methods

public View onCreateView(@NonNull LayoutInflater, ViewGroup container, Bundle savedInstanceState)
private void uploadImage();
private void getEnvironmentAnalysis();
private void setUI()

MainActivity
This activity class represents the first starting point feature before moving on to other feature fragments.
Attributes
private ActivityMainBinding binding;
Methods
protected void onCreate(Bundle savedInstanceState) private void createNotificationChannel()

3.2. Back-End Classes

Camera
This class represents the camera of the mobile device, it will provide the necessary pictures to be analysed
Attributes
camera picture: image
Methods
getImagefromCamera()

MachineLearningImageProcessing
This class contains the necessary machine learning and image processing to analyse the pictures and find distracting objects
Attributes
camera picture: image
Methods
findDistractingObjects(image) findShinyObjects(image) findColorScale(image)

Database Connection
This class represents the database connection for storing images in the Firebase Database.
Attributes
analysed picture: image
Methods
storeImage(image)

StoredData
This class represents the storage of analysed images and total activity count a user has to be displayed if necessary.
Attributes
analysed picture: image total counts: integer
Methods
getResults(image) getCount(integer)

ProfileTotalCounts
This class represents the total counts of all activity a user has to done and will be displayed on the profile.
Attributes
private int TotalNumberOfTasks private int TotalSessionsHour private int TotalNumberOfStudySessions
Methods
Getter and Setter Methods. private int displayNoOfTask() private int displaySessionNo() private int displayStudySesionNo()

Login/Register
This class represents login and register activities of user. The users will be able to login or register to the app from this class system.
Attributes
private string email private string password
Methods
Getter and Setter methods private void createAccount(email) private boolean validate(email, password) private void displayMessage() private void createUserID(email, password)

Authentication
This class contains the necessary authentication procedures to carry out safety and integrity.
Attributes
private String userID private String token
Methods
Getter and Setter methods. private boolean authenticateUser(userID, token)

ToDoList
This class represents the ToDoList feature attributes and methods.
Attributes
private int taskID private String content private Date date
Methods
private void deleteTaskService(taskID) private void addTaskService() private void updateTaskService(taskID, content, date)

ToDoListController
This class controls the actions that has been taken in the ToDoList class and executes them accordingly
Attributes
private int taskID private String content private Date date
Methods
private void addNewTaskController() private Task deleteTaskController(taskID) private Task updateTaskController(taskID, content, date) private void addDateController(taskID, date) private void addContentController(taskID, content) private Task getTaskCount()

4. Glossary

Pomodoro: It is a time management technique that involves working in intervals and taking breaks at specified intervals.

ADHD: It is a neurodevelopmental disorder, the cause of which has not been fully identified, but which usually begins in childhood and causes behaviors such as hyperactivity and lack of attention in people who have it.

Zone Check: In some parts of the report, the section called Zone Check indicates the feature in which we perform environmental analysis in our application.

5. References

- Centers for Disease Control and Prevention, “What is ADHD?,” *Centers for Disease Control and Prevention*, Sep. 27, 2023. https://www.cdc.gov/ncbddd/adhd/facts.html

- “Activity | Android Developers,” Android Developers, 2019. https://developer.android.com/reference/android/app/Activity