My Mental Health App

Group 9

Alexander, Yonatan	**SN: 251097927**
Kien Tang Tran, Michael	**SN: 250735158**
Aziz, Mohammad Yasin	**SN: 251282726**

Computer Science 4470A
The Department of Computer Science
The University of Western Ontario

Western

December 3, 2022

Please do NOT post our group's video to the course's YouTube Channel.

Contents

List of Figures

List of Tables

1 Problem Definition

Therapists and patients encounter many small yet persistent challenges during the therapeutic process concerning the patients' ability to self-manage between sessions.

While this project‘s scope is not large enough to address them all, the following stood out to us:

- Mood tracking between sessions.
  - Usually, at the start of a session therapists ask for an update about the time since the last session. However, self-reporting bias usually means that if the patient is having a bad day or is in a bad mood, their view on the time since the last session is skewed, sometimes heavily. Further, memory can be corrupted and events can easily be forgotten.
- Medication management
  - Inventory (to not run out of important medications, mood stabilizers are a good example in the context of mental health especially)
  - Persistence (taking medications as required and without missing a day)
- Symptoms
  - Medical (unrelated to the patients diagnosed)
- Other self care aspects, such as sleeping, eating, etc.

2 Project Objective

O1 Patients can add new entries.

O2 Patients can view existing entries.

O3 Patients can delete existing entries.

O4 User will be able to view the entries from the past week, at a glance, in a dedicated view.

O5 Learning the process of project development through application of knowledge acquired during the Computer Science 4471A Course.

O6 The creation of an intuitive and user friendly self-management mobile app (Android).

3 Stakeholders List

- Patients - Internal stakeholder

3.1 Internal Stakeholders

Therapists - External stakeholder

external - anyone who has an influence including people who write certain libraries and so on (include instructor and TA as external stakeholders)

internal - programmers, maintainable, customers, users, etc [double check this point's list]

1. Patients (main users and the target audience).

3.2 External Stakeholders

1. Therapists
2. Legislation and public policy (for example security of medical information)
3. Governing bodies (such as the CRPO - The College of Registered Psychotherapists of Ontario)
4. Instructor
5. TA
6. Libraries:
   (a) Android.
   (b) Google.
   (c) MPAndroid (to create the graphs).
   (d) JetBrains (makers of Android Studio).
   (e) Java Libraries

4 Success/Acceptance Criteria for each Stakeholder

- As a user, I want to be able to store my mood on a particular day, on a scale of 0 to 10.
- As a user, I want to be able to store my medication details such as the medication, dosage, etc.
- As a user, I want to be able to store my symptom details on a particular day.
- As a user, I want to be able to view my past mood reports for my therapy visit.
- As a user, I want to be able to view my past symptoms for my therapy visit.
- As a user, I want to be able to view my past medications for my therapy visit.

5 Use case diagram(s)

5.1 Use case Diagram 1: Mood

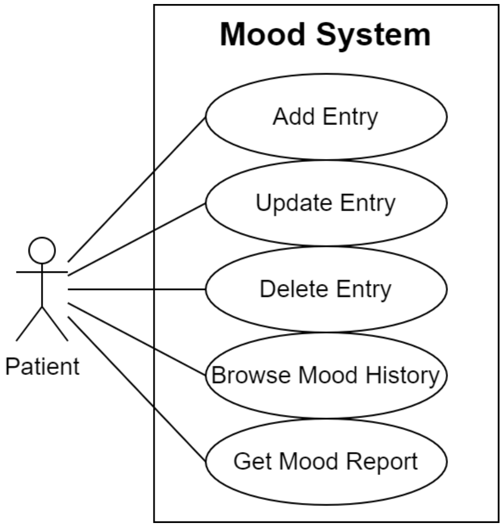


Figure 1: Use case Diagram 1: The Mood Subsystem

Contains the Mood subsystem.

5.2 Use case Diagram 2: Medication

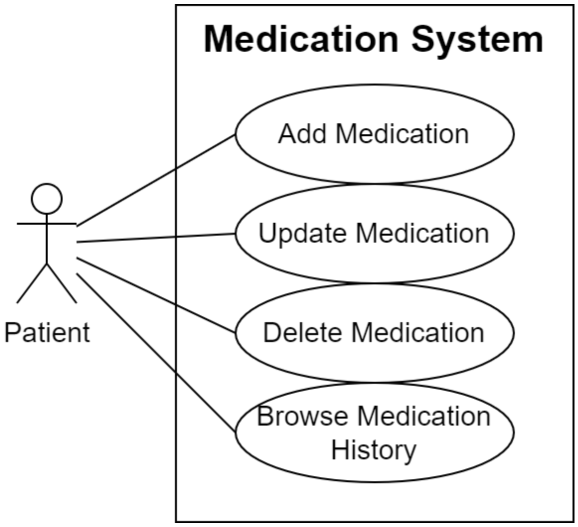


Figure 2: Use case Diagram 2: The Medication Subsystem

Contains the Medication subsystem.

5.3 Use case Diagram 3: Symptom

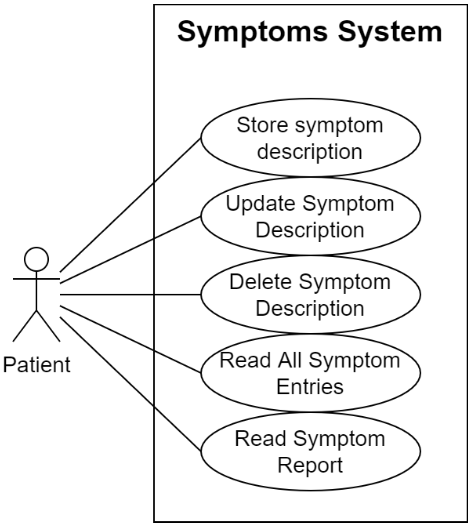

Symptoms System
Store symptom description
Update Symptom Description
Delete Symptom Description
Read All Symptom Entries
Read Symptom Report
Patient

Figure 3: Use case Diagram 3: The Symptom Subsystem

Contains the Symptom subsystem.

6 Use case Descriptions

6.1 Use case Description: Add Medication

Use Case Name: Add Medication
Scenario: Add medication entry to database.
Brief Description: When a Patient (User) enters information for a new medication entry and presses “Submit”, add the information to the medications’ table in the database.
Triggering Event: Patient (User) enters the “Medication” view.
Actors: Patient (User)
Related Use cases: Update Medication, Delete Medication, Browse Medication History
Stakeholders: Patient
Preconditions: App is open on the “Medications” Screen.
PostConditions: A new entry was added to the medications’ table in the database.
Flow of Activities:

Actor	**System**
1. User opens “Medications” screen	
2. User fills the “Enter the medication name” field	
3. (Optional) User fills the “Enter the brand name” field	
4. (Optional) User fills the “Enter the medication quantity” field	
5. (Optional) User fills the “Enter the medication quantity unit” field	
6. (Optional) User fills the “Enter the medication frequency” field	
7. User pressed “Add Medication Entry”	7.1 System confirms step 2 was completed 7.2 System updates the database with new entry

Table 1: Use case Description: Add Medication, Flow of Activities

Exception conditions:

The user did not fulfill step 2.

6.2 Use case Description: Browse Medication History

Use Case Name: Browse Medication History
Scenario: The user wants to view the medication entries.
Brief Description: When a Patient (User) enters opens the “Medication List” view, the list of medications should appear.
Triggering Event: Patient (User) enters the “Medication List” view.
Actors: Patient (User)
Related Use cases: Update Medication, Delete Medication, Add Medication
Stakeholders: Patient

Preconditions: App is open on any non-“Medications List” Screen.
PostConditions: None (can be empty).
Flow of Activities:

Actor	**System**
1. User opens the “Medication List” screen	1.1 The system propagates past entries into the view (if there are any).

Table 2: Use case Description: Browse Medication History, Flow of Activities

7 Sequence diagram(s) for the selected use case for descriptions

7.1 Sequence Diagram 1: Add Medication

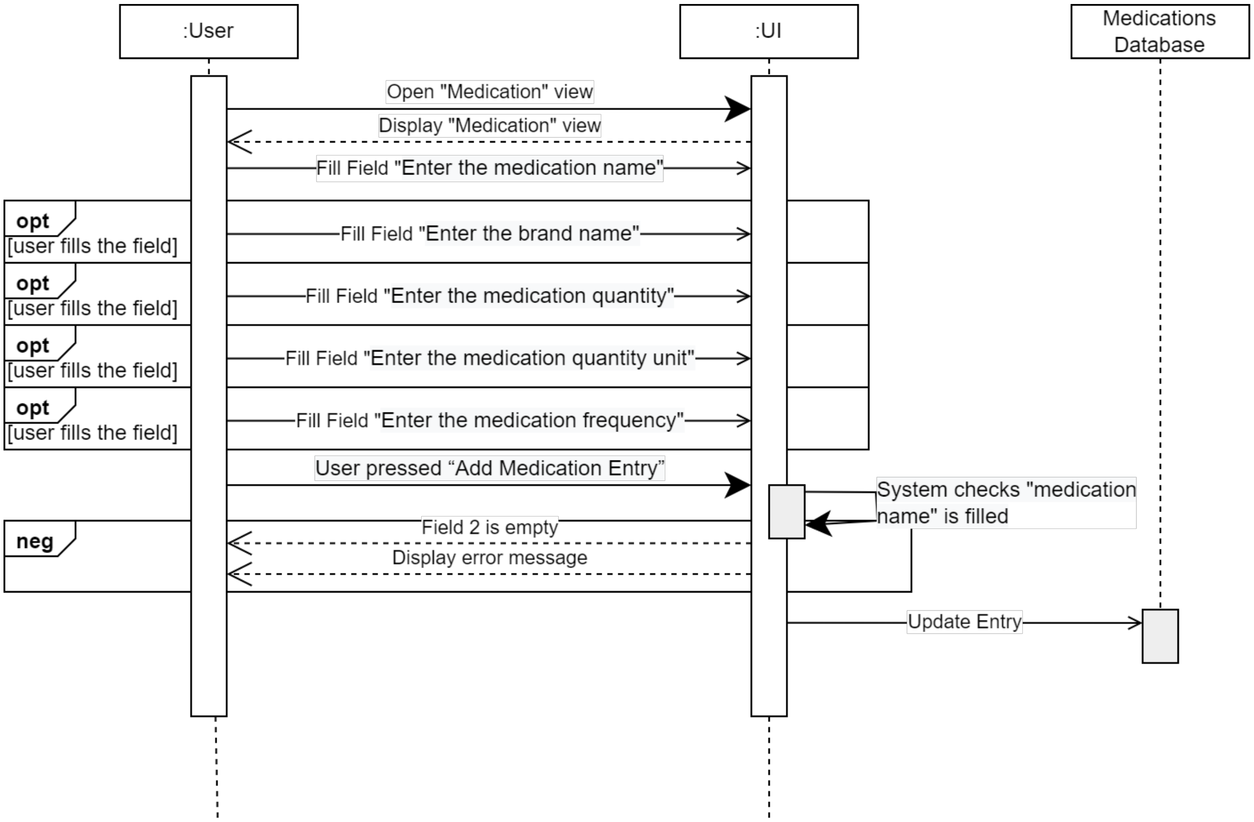

:User
:UI
Medications Database
Open "Medication" view
Display "Medication" view
Fill Field "Enter the medication name"
opt
[user fills the field]
Fill Field "Enter the brand name"
opt
[user fills the field]
Fill Field "Enter the medication quantity"
opt
[user fills the field]
Fill Field "Enter the medication quantity unit"
opt
[user fills the field]
Fill Field "Enter the medication frequency"
User pressed “Add Medication Entry”
System checks "medication name" is filled
neg
Field 2 is empty
Display error message
Update Entry

Figure 4: Class Diagram: Relationships.

7.2 Sequence Diagram 2: Browse Medication History

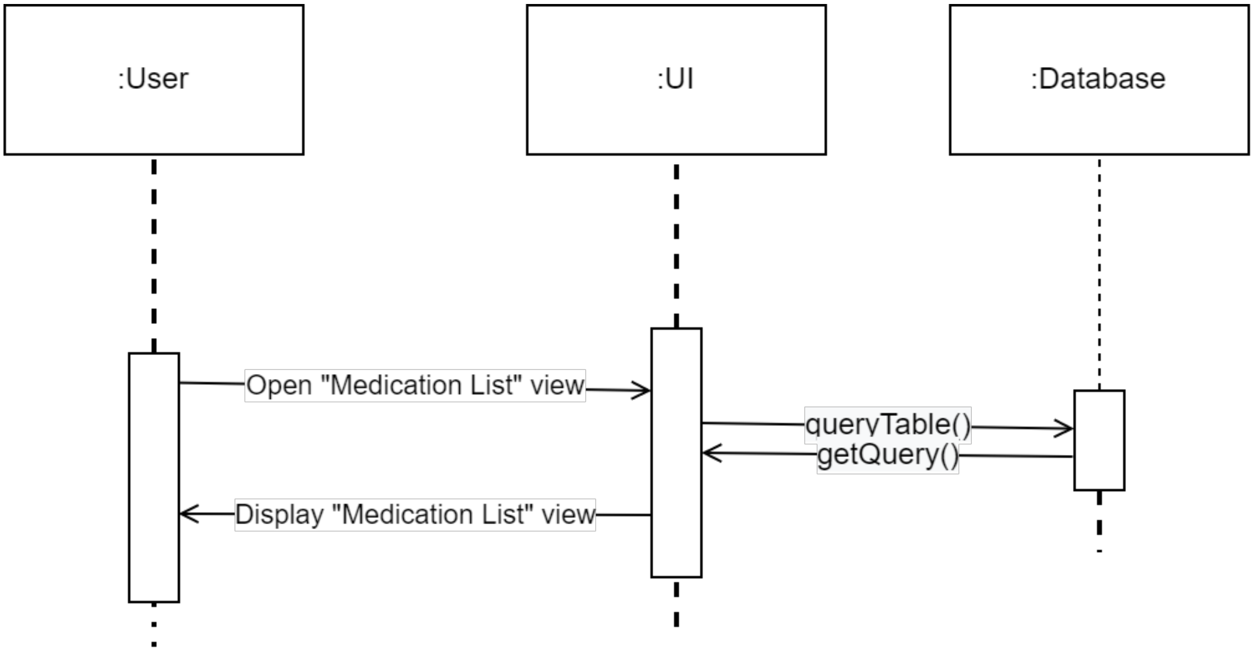


Figure 5: Class Diagram: Relationships.

8 System Architecture

Unfortunately, writing the code in Java meant that our chosen architecture (Model-View-ViewModel) was not feasible, resulting in needing to change the System Architecture to Model View Controller very late in development.

System Architecture: Model View Controller.

9 Detailed Class diagram(s)

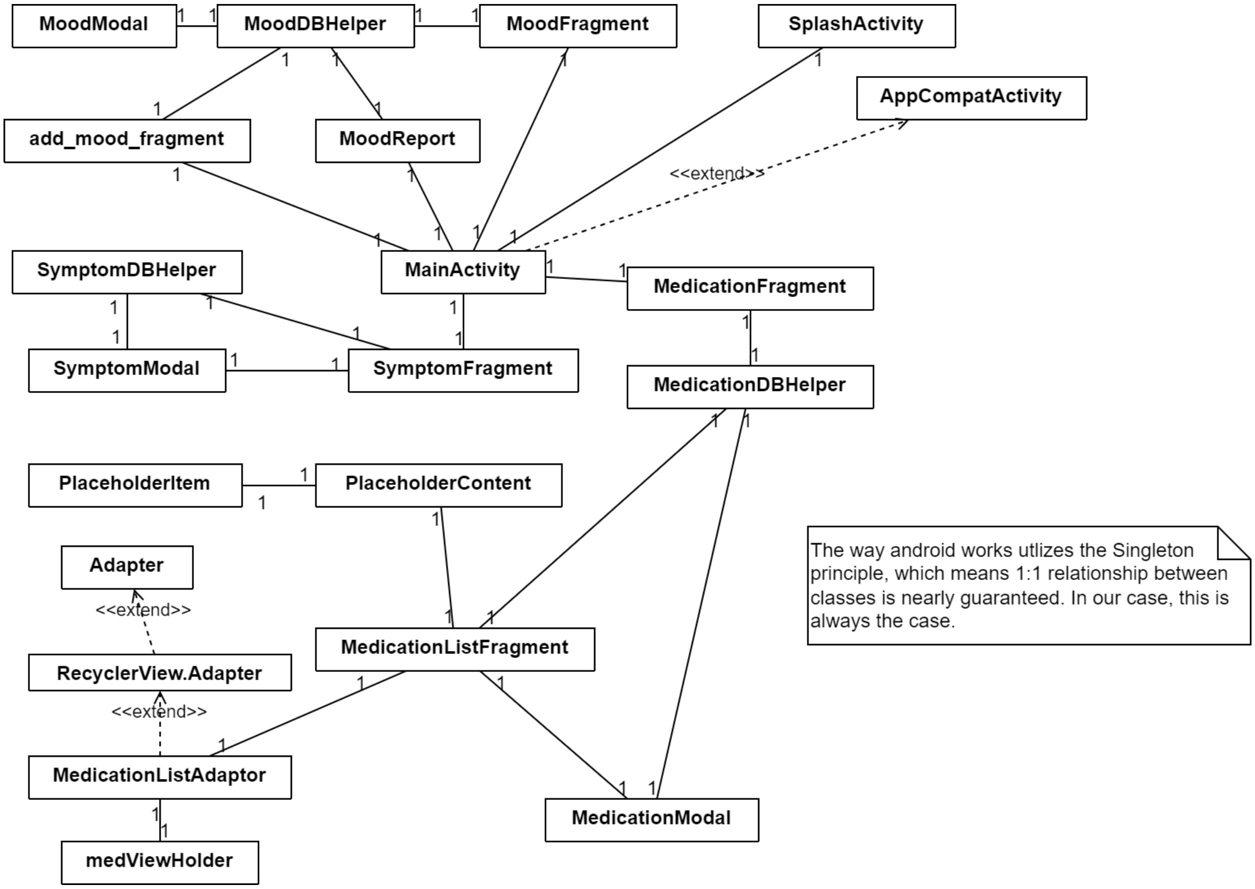


Figure 6: Class Diagram: Relationships.

Contains **all** classes in the system, and the relationships between them. The other class diagrams contain the respective methods.

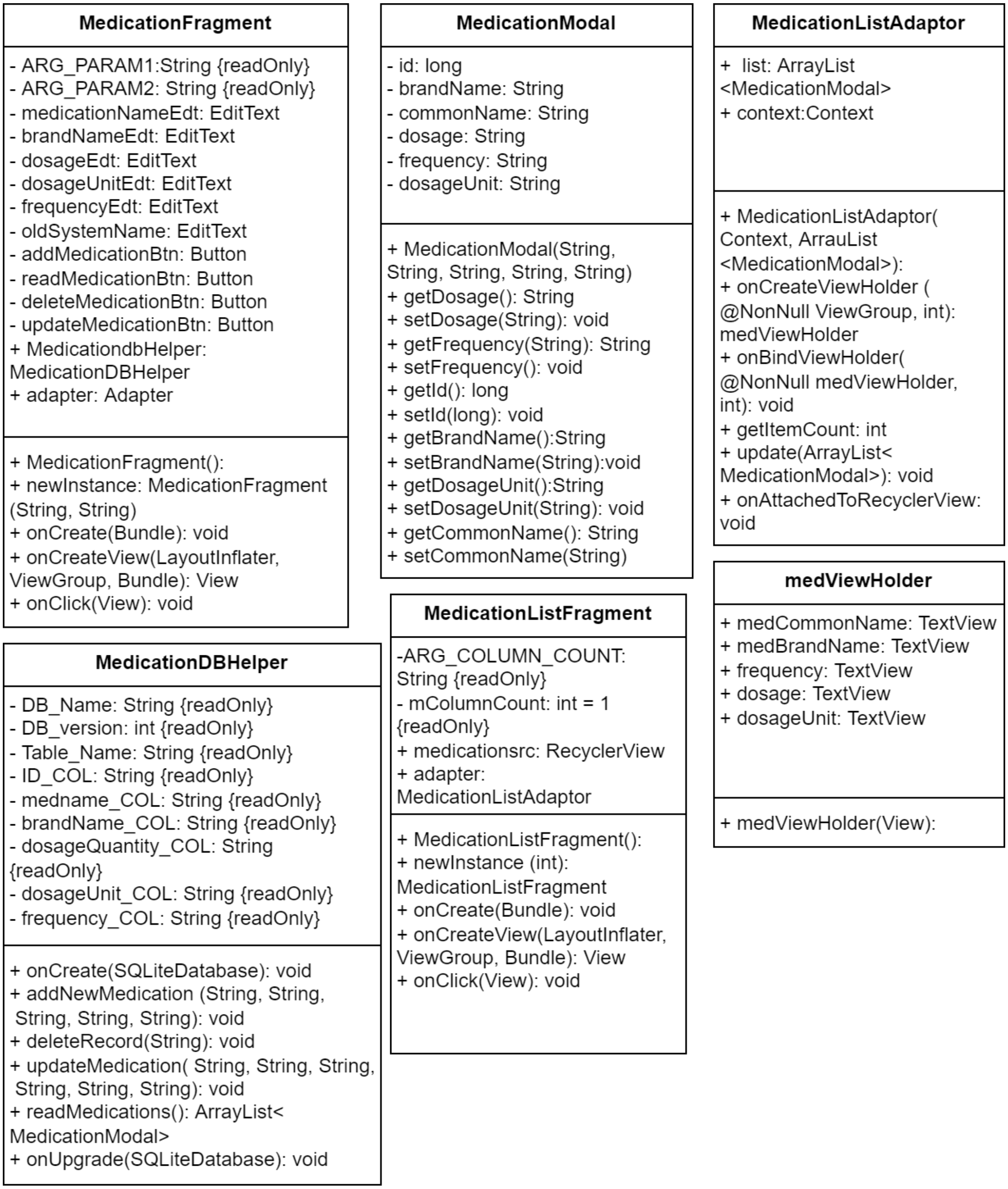
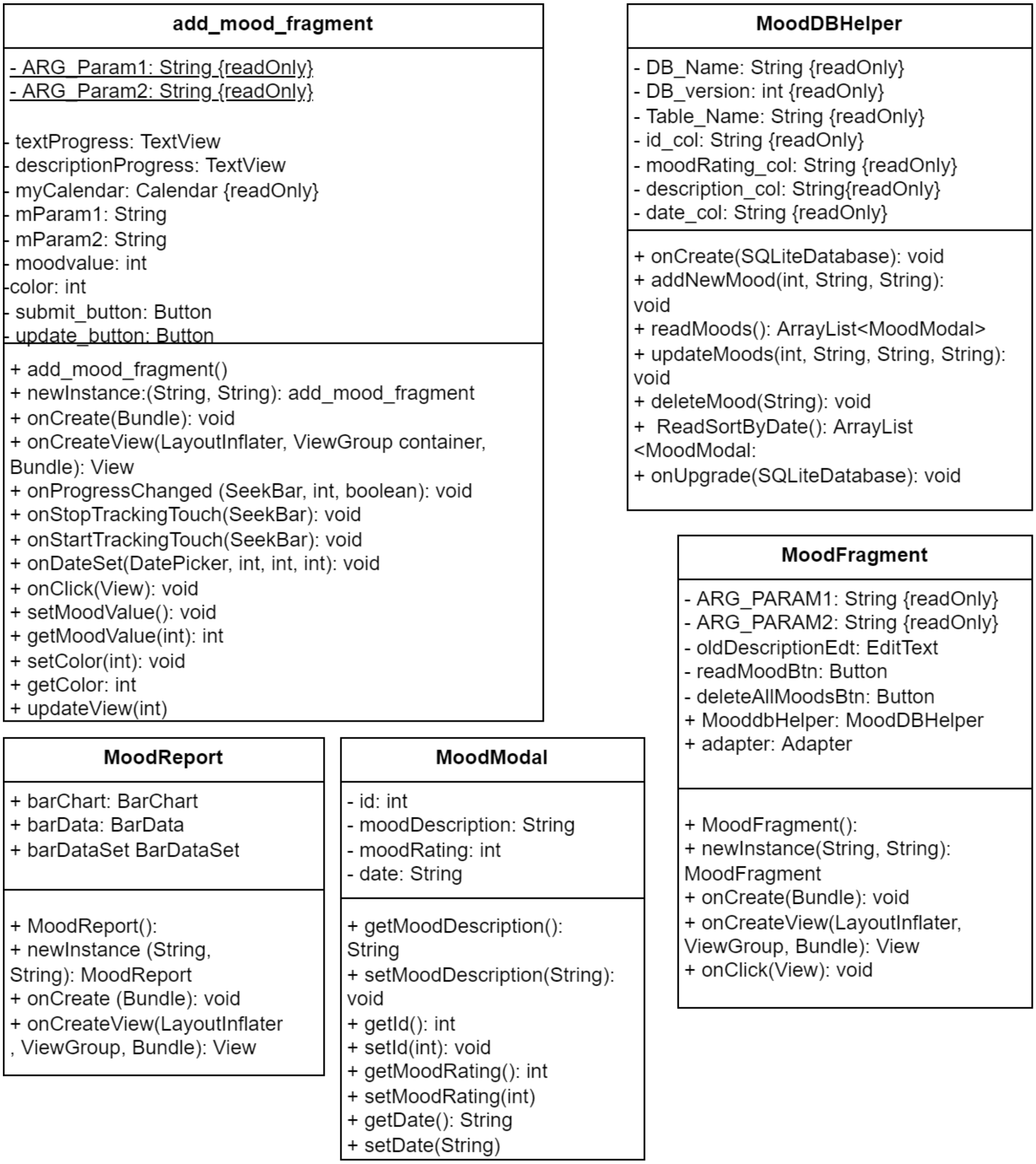

MedicationFragment
- ARG_PARAM1:String {readOnly}
- ARG_PARAM2: String {readOnly}
- medicationNameEdt: EditText
- brandNameEdt: EditText
- dosageEdt: EditText
- dosageUnitEdt: EditText
- frequencyEdt: EditText
- oldSystemName: EditText
- addMedicationBtn: Button
- readMedicationBtn: Button
- deleteMedicationBtn: Button
- updateMedicationBtn: Button
+ MedicationdbHelper: MedicationDBHelper
+ adapter: Adapter
+ MedicationFragment():
+ newInstance: MedicationFragment (String, String)
+ onCreate(Bundle): void
+ onCreateView(LayoutInflater, ViewGroup, Bundle): View
+ onClick(View): void
MedicationModal
- id: long
- brandName: String
- commonName: String
- dosage: String
- frequency: String
- dosageUnit: String
+ MedicationModal(String, String, String, String, String)
+ getDosage(): String
+ setDosage(String): void
+ getFrequency(String): String
+ setFrequency(): void
+ getId(): long
+ setId(long): void
+ getBrandName():String
+ setBrandName(String):void
+ getDosageUnit():String
+ setDosageUnit(String): void
+ getCommonName(): String
+ setCommonName(String)
MedicationListAdaptor
+ list: ArrayList <MedicationModal>
+ context:Context
+ MedicationListAdaptor(Context, ArrauList <MedicationModal>):
+ onCreateViewHolder (@NonNull ViewGroup, int): medViewHolder
+ onBindViewHolder(@NonNull medViewHolder, int): void
+ getItemCount: int
+ update(ArrayList< MedicationModal>): void
+ onAttachedToRecyclerView: void
medViewHolder
+ medCommonName: TextView
+ medBrandName: TextView
+ frequency: TextView
+ dosage: TextView
+ dosageUnit: TextView
+ medViewHolder(View):
MedicationDBHelper
- DB_Name: String {readOnly}
- DB_version: int {readOnly}
- Table_Name: String {readOnly}
- ID_COL: String {readOnly}
- medname_COL: String {readOnly}
- brandName_COL: String {readOnly}
- dosageQuantity_COL: String {readOnly}
- dosageUnit_COL: String {readOnly}
- frequency_COL: String {readOnly}
+ onCreate(SQLiteDatabase): void
+ addNewMedication (String, String, String, String, String): void
+ deleteRecord(String): void
+ updateMedication(String, String, String, String, String, String): void
+ readMedications(): ArrayList< MedicationModal>
+ onUpgrade(SQLiteDatabase): void
MedicationListFragment
-ARG_COLUMN_COUNT: String {readOnly}
- mColumnCount: int = 1 {readOnly}
+ medicationsrc: RecyclerView
+ adapter: MedicationListAdaptor
+ MedicationListFragment():
+ newInstance (int): MedicationListFragment
+ onCreate(Bundle): void
+ onCreateView(LayoutInflater, ViewGroup, Bundle): View
+ onClick(View): void

Figure 7: Class Diagram: Medication

Contains **all** classes relating directly Medications. Relationships included only as part of *Class Diagram: Medication*.



```
add_mood_fragment
- ARG_Param1: String {readOnly}
- ARG_Param2: String {readOnly}

- textProgress: TextView
- descriptionProgress: TextView
- myCalendar: Calendar {readOnly}
- mParam1: String
- mParam2: String
- moodvalue: int
-color: int
- submit_button: Button
- update_button: Button
+ add_mood_fragment()
+ newInstance:(String, String): add_mood_fragment
+ onCreate(Bundle): void
+ onCreateView(LayoutInflater, ViewGroup container,
Bundle): View
+ onProgressChanged (SeekBar, int, boolean): void
+ onStopTrackingTouch(SeekBar): void
+ onStartTrackingTouch(SeekBar): void
+ onDateSet(DatePicker, int, int, int): void
+ onClick(View): void
+ setMoodValue(): void
+ getMoodValue(int): int
+ setColor(int): void
+ getColor: int
+ updateView(int)
```

```
MoodDBHelper
- DB_Name: String {readOnly}
- DB_version: int {readOnly}
- Table_Name: String {readOnly}
- id_col: String {readOnly}
- moodRating_col: String {readOnly}
- description_col: String{readOnly}
- date_col: String {readOnly}
+ onCreate(SQLiteDatabase): void
+ addNewMood(int, String, String):
void
+ readMoods(): ArrayList<MoodModal>
+ updateMoods(int, String, String, String):
void
+ deleteMood(String): void
+  ReadSortByDate(): ArrayList
<MoodModal:
+ onUpgrade(SQLiteDatabase): void
```

```
MoodFragment
- ARG_PARAM1: String {readOnly}
- ARG_PARAM2: String {readOnly}
- oldDescriptionEdt: EditText
- readMoodBtn: Button
- deleteAllMoodsBtn: Button
+ MooddbHelper: MoodDBHelper
+ adapter: Adapter

+ MoodFragment():
+ newInstance(String, String):
MoodFragment
+ onCreate(Bundle): void
+ onCreateView(LayoutInflater,
ViewGroup, Bundle): View
+ onClick(View): void
```

```
MoodReport
+ barChart: BarChart
+ barData: BarData
+ barDataSet BarDataSet

+ MoodReport():
+ newInstance (String,
String): MoodReport
+ onCreate (Bundle): void
+ onCreateView(LayoutInflater
, ViewGroup, Bundle): View
```

```
MoodModal
- id: int
- moodDescription: String
- moodRating: int
- date: String
+ getMoodDescription():
String
+ setMoodDescription(String):
void
+ getId(): int
+ setId(int): void
+ getMoodRating(): int
+ setMoodRating(int)
+ getDate(): String
+ setDate(String)
```

Figure 8: Class Diagram: Mood

Contains **all** classes relating directly Mood. Relationships included only as part of *Class Diagram: Medication*.

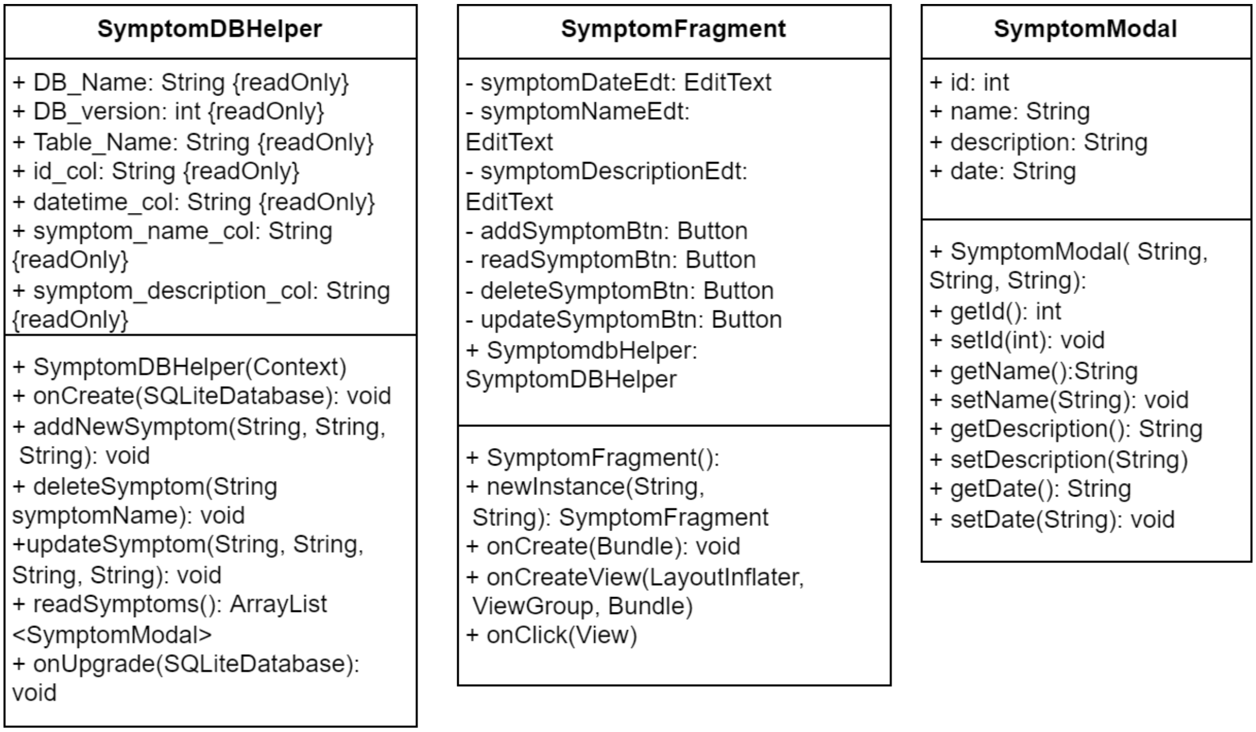

SymptomDBHelper
+ DB_Name: String {readOnly}
+ DB_version: int {readOnly}
+ Table_Name: String {readOnly}
+ id_col: String {readOnly}
+ datetime_col: String {readOnly}
+ symptom_name_col: String {readOnly}
+ symptom_description_col: String {readOnly}
+ SymptomDBHelper(Context)
+ onCreate(SQLiteDatabase): void
+ addNewSymptom(String, String, String): void
+ deleteSymptom(String symptomName): void
+updateSymptom(String, String, String, String): void
+ readSymptoms(): ArrayList <SymptomModal>
+ onUpgrade(SQLiteDatabase): void
SymptomFragment
- symptomDateEdt: EditText
- symptomNameEdt: EditText
- symptomDescriptionEdt: EditText
- addSymptomBtn: Button
- readSymptomBtn: Button
- deleteSymptomBtn: Button
- updateSymptomBtn: Button
+ SymptomdbHelper: SymptomDBHelper
+ SymptomFragment():
+ newInstance(String, String): SymptomFragment
+ onCreate(Bundle): void
+ onCreateView(LayoutInflater, ViewGroup, Bundle)
+ onClick(View)
SymptomModal
+ id: int
+ name: String
+ description: String
+ date: String
+ SymptomModal(String, String, String):
+ getId(): int
+ setId(int): void
+ getName():String
+ setName(String): void
+ getDescription(): String
+ setDescription(String)
+ getDate(): String
+ setDate(String): void

Figure 9: Class Diagram: Symptoms

Contains **all** classes relating directly Symptoms. Relationships included only as part of *Class Diagram: Medication*.

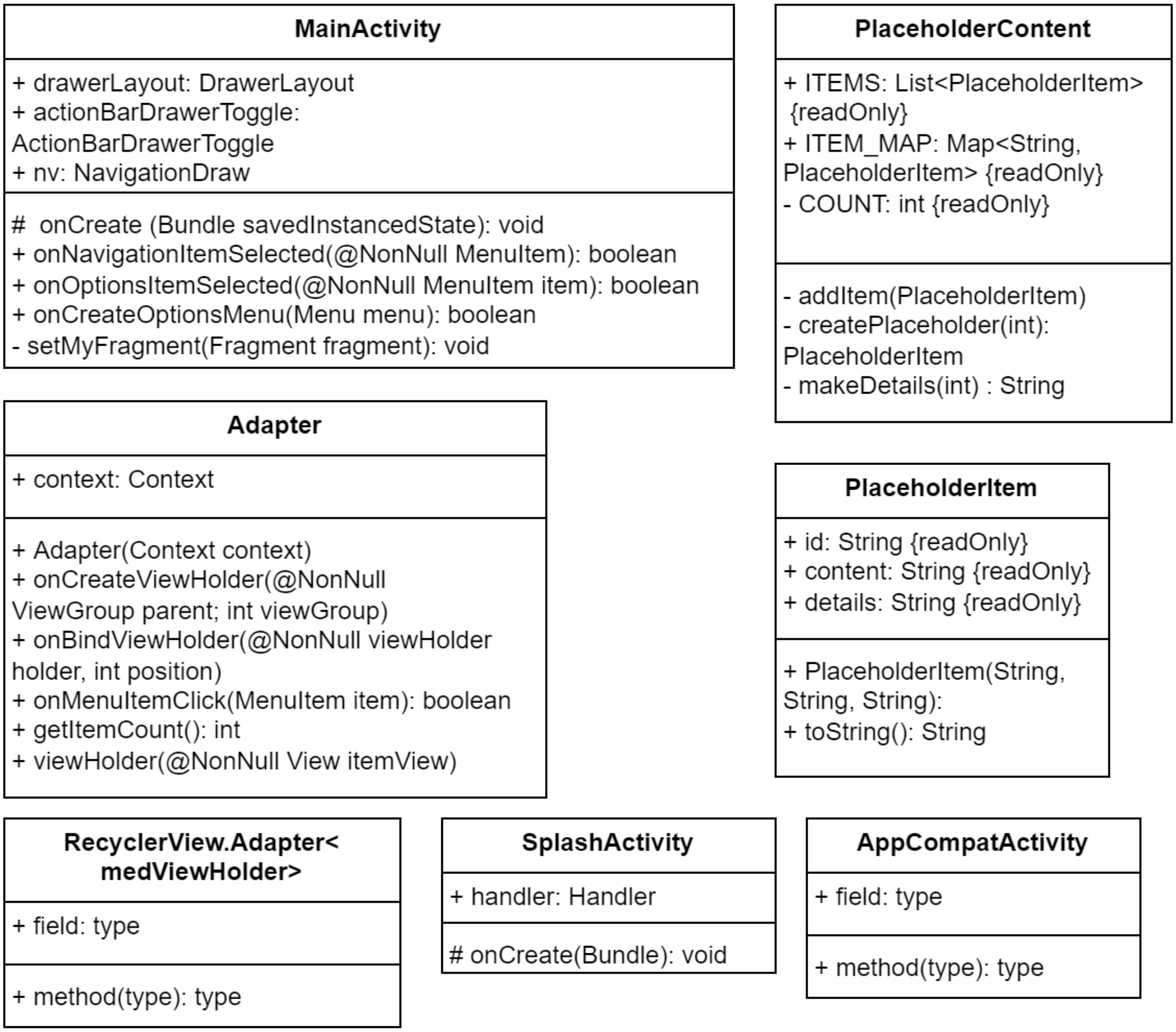

MainActivity
+ drawerLayout: DrawerLayout
+ actionBarDrawerToggle: ActionBarDrawerToggle
+ nv: NavigationDraw
# onCreate (Bundle savedInstancedState): void
+ onNavigationItemSelected(@NonNull MenuItem): boolean
+ onOptionsItemSelected(@NonNull MenuItem item): boolean
+ onCreateOptionsMenu(Menu menu): boolean
- setMyFragment(Fragment fragment): void
PlaceholderContent
+ ITEMS: List<PlaceholderItem> {readOnly}
+ ITEM_MAP: Map<String, PlaceholderItem> {readOnly}
- COUNT: int {readOnly}
- addItem(PlaceholderItem)
- createPlaceholder(int): PlaceholderItem
- makeDetails(int) : String
Adapter
+ context: Context
+ Adapter(Context context)
+ onCreateViewHolder(@NonNull ViewGroup parent; int viewGroup)
+ onBindViewHolder(@NonNull viewHolder holder, int position)
+ onMenuItemClick(MenuItem item): boolean
+ getItemCount(): int
+ viewHolder(@NonNull View itemView)
PlaceholderItem
+ id: String {readOnly}
+ content: String {readOnly}
+ details: String {readOnly}
+ PlaceholderItem(String, String, String):
+ toString(): String
RecyclerView.Adapter< medViewHolder>
+ field: type
+ method(type): type
SplashActivity
+ handler: Handler
# onCreate(Bundle): void
AppCompatActivity
+ field: type
+ method(type): type

Figure 10: Class Diagram: Other

Contains all classes **not** already included in the other diagrams above. Relationships included only as part of *Class Diagram: Medication.*

10 State-machine diagram

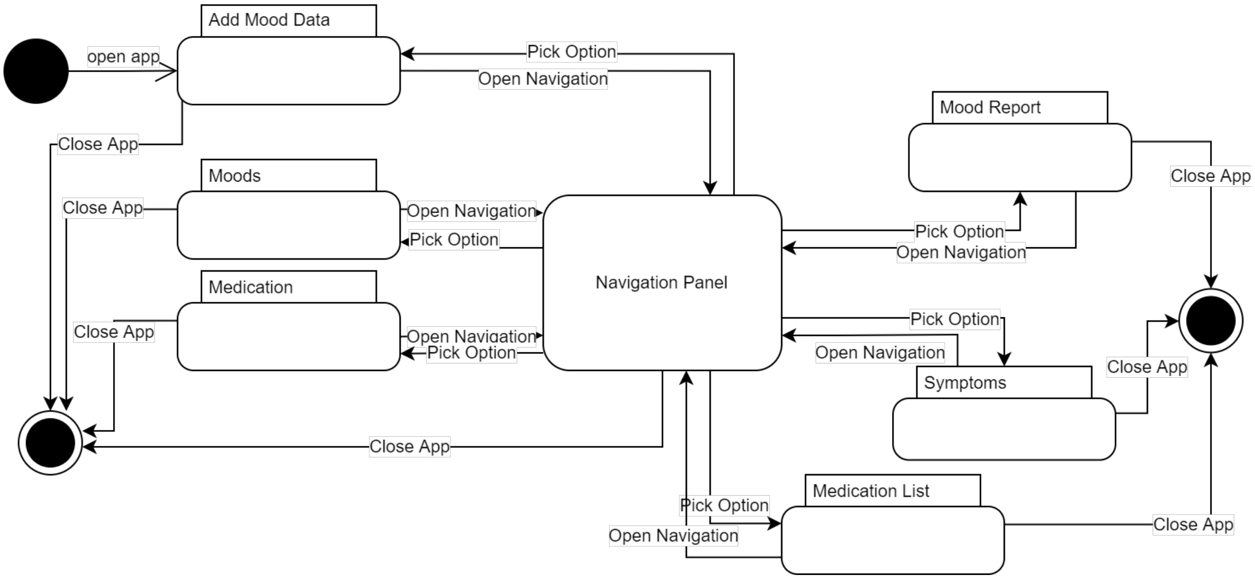


Figure 11: State Diagram: Whole System

Contains the state diagram of the whole system. Each specific screen in the UI interface being its own composite state. Each composite state has substrates, however they all follow the user interaction → system reaction pattern.

11 Entity Relationship Diagram (Data modelling)

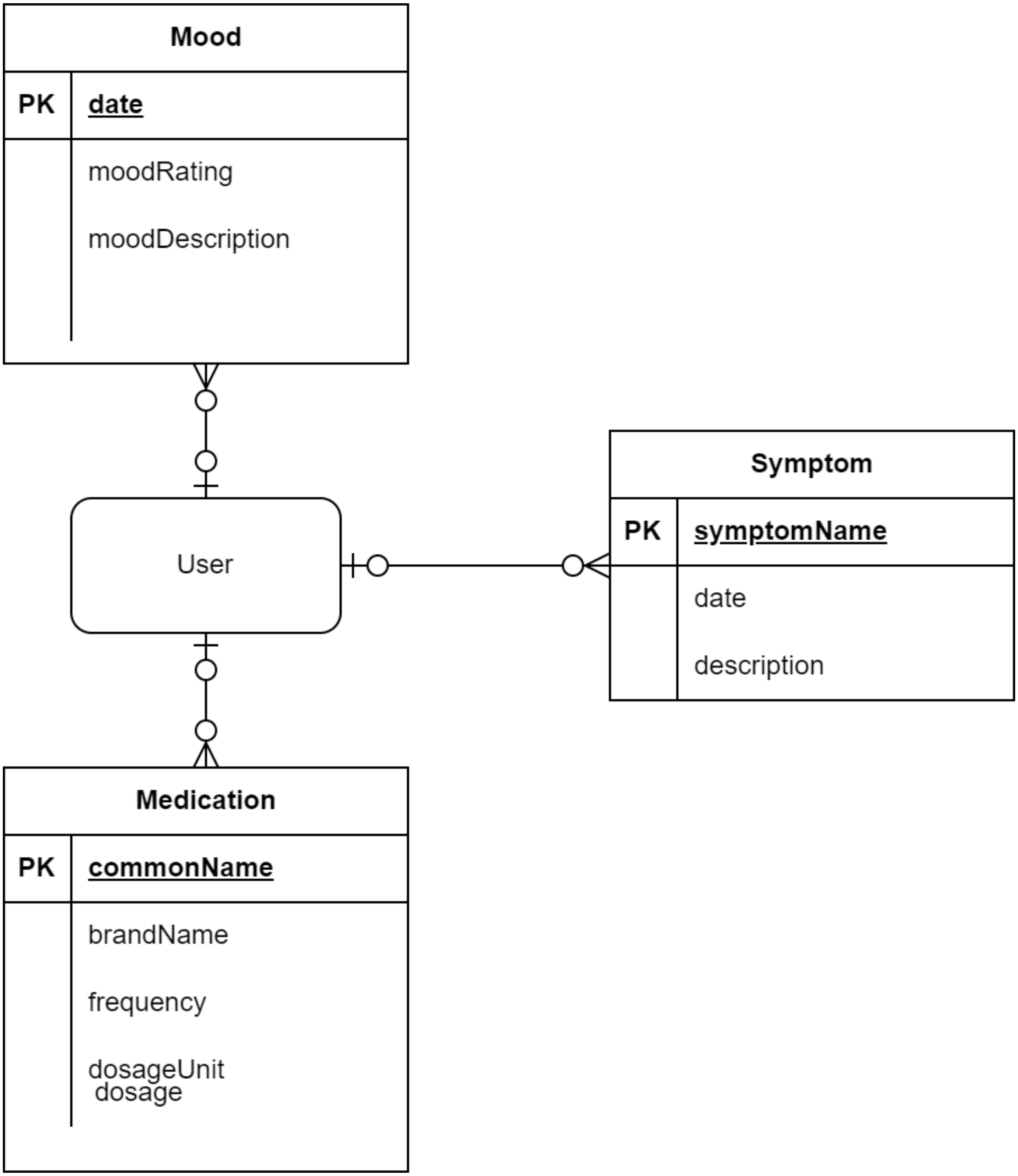


Figure 12: Entity Relationship Diagram

The entity relationship diagram (Data modelling) for the system.

12 GitHub Link

You MUST ask Yonatan Alexander for access, as this repository is set to private.

https://github.com/Jon-AL/4471—Therapy-Support-App

13 Conclusion

In many ways, the project did not go according to plan. We made a number of mistakes, but we believe the biggest was not making the project in Kotlin. Making the project in Kotlin would have allowed us to make the app's UI programmatically, which would have been a huge improvement. Additionally, this project turned out to be far larger in scope than we anticipated, simply because we were learning as we went.

In summary, many mistakes were made, but making such mistakes in a project where the intention is learning is exactly the right time to make them - and therefore we believe that those mistakes were (overall, at least) a positive.

14 References

1. **MyTherapy (app):** Your personal pill reminder and medication tracker app[**?**]
2. **Bipolar UK's Mood Tracker (app):** Our new Mood Tracker app can make it much easier to record your daily mood, medications, emotions and how much sleep you've had[**?**]

main

Appendices

A Project WBS and Gantt Chart

A.1 Project WBS

Spread over a number of pages due to Project Libre's many limitations

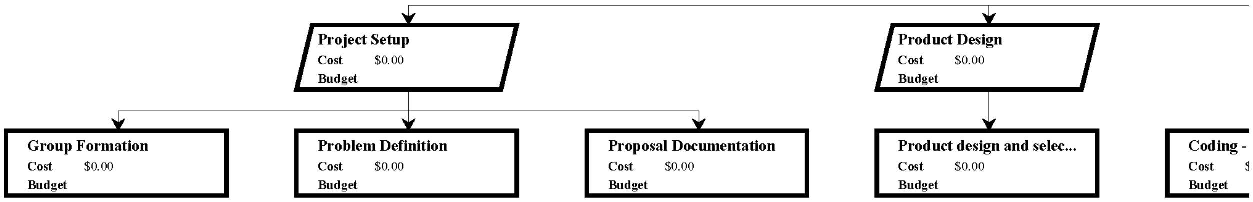


Figure 13: WBS Part 1 out of 5

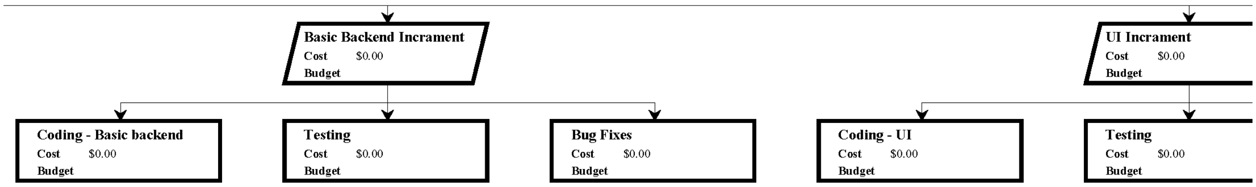


Figure 14: WBS Part 2 out of 5

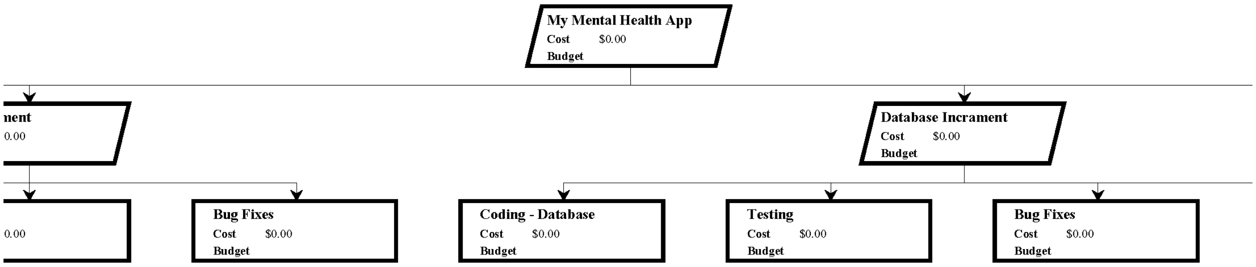


Figure 15: WBS Part 3 out of 5

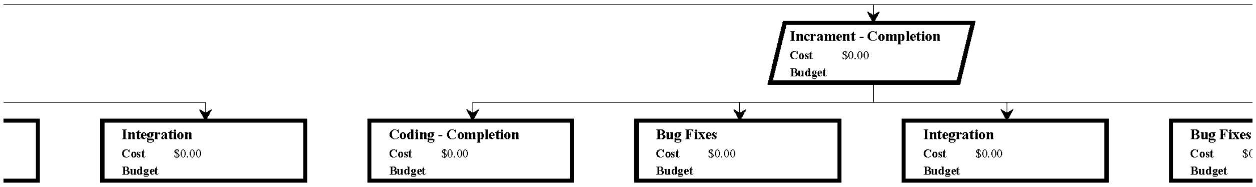


Figure 16: WBS Part 4 out of 5

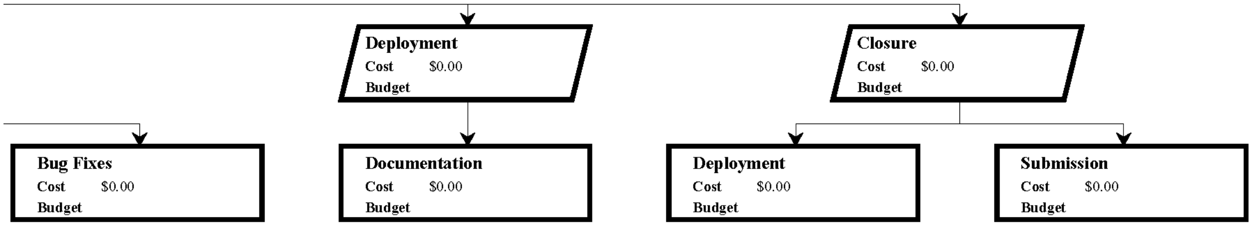


Figure 17: WBS Part 5 out of 5

A.2 Project Gantt Chart

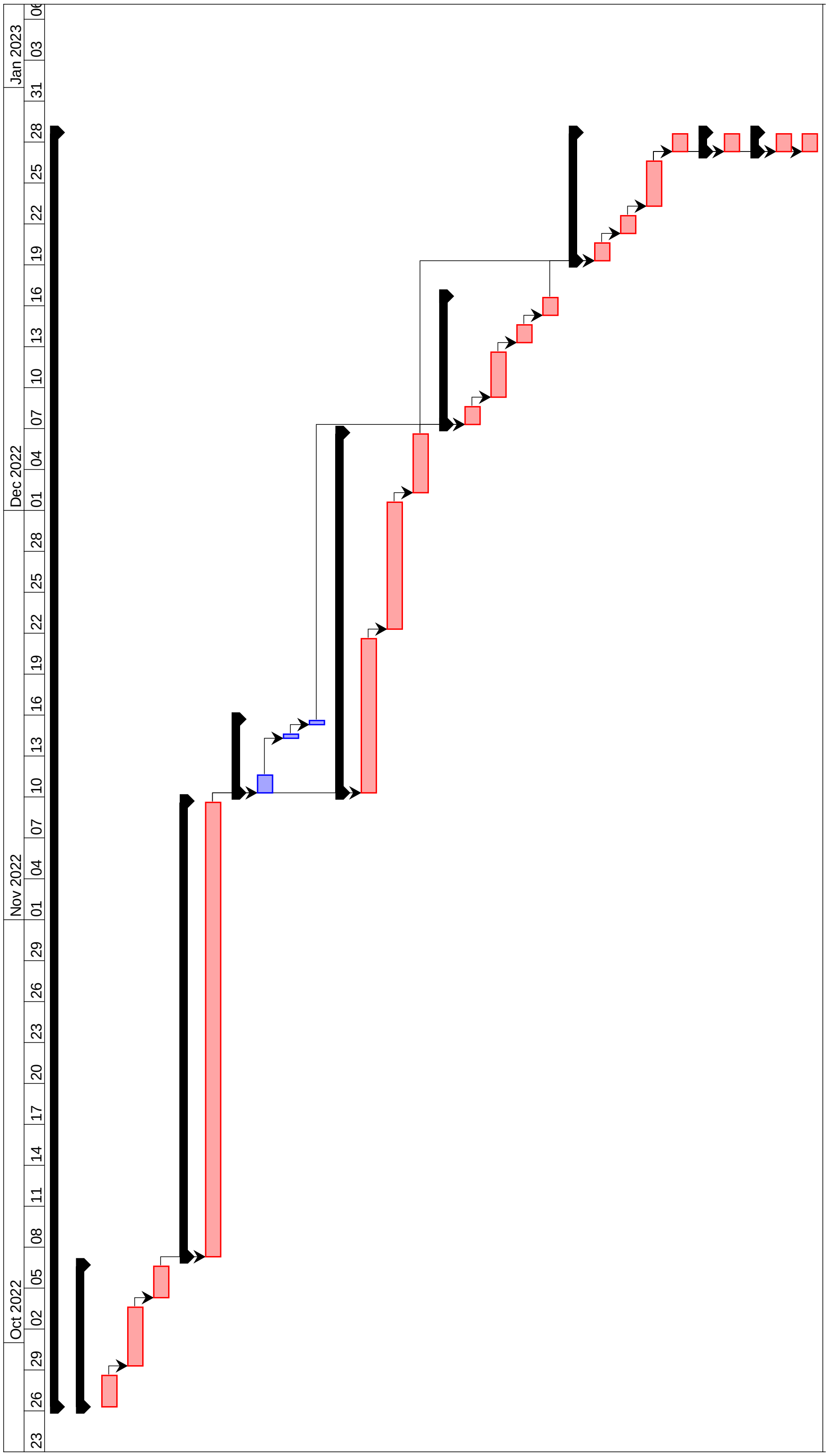


Figure 18: Gantt Chart

B Task Assignment Matrix

Task	**Task Owner**	**Support**
Basic UI User Interfaces	Yassine	Yonatan, Michael
Advanced UI User Interfaces	Michael	Yonatan
Front end (UI) completion	Michael	Yonatan
WBS	Yonatan, Michael	
Gantt Chart	Michael	Yonatan
Task Assignment Matrix	Yonatan, Michael	
Testing Matrix	Yonatan, Michael	
User Stories for Project UI	Yassine	Yonatan, Michael
Skeleton Code	Yonatan	Michael, Yassine
Database Integration with Backend	Yonatan, Michael	
Database Execution	Michael	Yonatan
Front end Integration with Backend	Michael	Yonatan
Use Case Diagrams	Yonatan, Michael	
Use Case Descriptions	Michael	Yonatan
Sequence Diagrams	Yonatan, Michael	
Class Diagram	Yonatan, Michael	
Data Entity Relationship Diagram	Yonatan, Michael	
State Diagrams	Michael	Yonatan
Video	Yonatan	Michael
Slideshow	Michael, Yonatan	
Report Editing	Yonatan	Michael

Table 3: Task Assignment Matrix

This table reflects what happened in practice, not what the plan was initially.

C Sample of commits on the selected version control system

D Minutes

D.1 26-Sep-'22

Present:

- Yonatan
- Michael

Not Present:

- Yasin

Discussed:

- Development proposal and development details.
- The project will be a therapy application that helps users gather, store and present their information to be used by patients.
- Suggested so far:
  - Incremental development.
  - Smartphone app.
  - Web app.
- Suggested Programming languages:
  - Dart.
  - Python.
  - Javascript.
  - C++.
  - Java.
- Stakeholders:
  - Clients.
  - Therapists and other mental health professionals.
- Features:
  - Mood self-reported values.
  - Medication tracking.

D.2 30-Sep-'22

Present:

- Yonatan
- Michael

Not Present:

- Yasin

Discussed:

- Project proposal was accepted.
- Instructor mentioned a number of points during lecture in the context of our project:
  - Patient data requirements with respect to privacy.
  - Identifying the stakeholders with who should be able to access data.
  - We need to identify the use cases carefully of what we are able to do.
  - We have to be able to clearly define the project objective and description.
- An aggressive timeline has been suggested to keep with coursework. This timeline is subject to change due to other courses requiring our attention.
- Github was setup by Yonatan.
- Report is in progress.

D.3 04-Oct-'22 and 05-Oct-'22

Discussed:

A timeline has been suggested to be finishing up to the user acceptance criteria. QT has been suggested as a platform to build an Android app.

The report has been started in LaTex by Yonatan. Proposed statements will be made by the group once we reviewed the work.

D.4 14-Oct-'22

Present:

- Yonatan
- Michael
- Yasin

Discussed:

Debate about the code is that we have no firm agreements on project libraries and frameworks. Suggestions include: salesforce as the backend. Flutter or ReactNative has also been suggested. Qt.

D.5 21-Oct-'22 to 28-Oct-'22

A lot of project will be developed on the reading week. Professor wants us to use Docker, and create a Gantt chart. He explicitly states that he wants us to assign backup roles to our task assignment matrix.

D.6 01-Nov-'22

Present:

- Yonatan
- Michael

Not Present:

- Yasin

Discussed:

Planned architecture:

  - repository style for the databases.
  - Event-driven for reminders.
  - Component-based.

- Today's objective:
  - Architecture and design patterns were discussed.
  - Get the backend done behind the scenes.
  - Another meeting: 9:30 am to 11:30 am tomorrow (November 2nd, 2022), want to start writing code.

D.7 02-Nov-'22

Present:

- Yonatan
- Michael

Not Present:

- Yasin

Discussed:

- We are switching the development process from incremental to a hybrid agile-iterative method.
- Scope and requirements changed significantly throughout the process.
- We are also incorporating agile methods to respond to changes over time.
- Kanban board has been created inside GitHub repository.
- User stories will be created for UI to make the interface clearer to write components and understand the problems.
- Drafts have been started but significant changes need to be created.
- An Android application will be created.
- The data will be stored locally with some encryption to get us started.

- Yonatan wants to push simple skeleton code, will require approval from Michael and Yassine.
- We want to finish baseline features executed and finished by November 6th.
- The list of things we want executed:
  - A basic UI - covered by Yassine (done).
  - Basic backend - covered by Yonatan
  - Database integration - covered by Michael
- We will need significant research need before implementations.
- No meeting will be scheduled for tomorrow.
- A few mini-meetings between Friday to Sunday.

D.8 02-Nov-'22

Present:

- Yonatan
- Michael

Not Present:

- Yasin

Discussed:

- Backend talk
- Significant modifications need to be executed for medication.
- We will have to split this into 3 main Tables:
  1. Medication info
  2. Medication inventory history
  3. Medication intake history
- Mood will be generated next.
- Symptoms will be generated last.
- Backend sample is expected to be done by Sunday late afternoon.
- Yassine will present a frontend demo on November 6th.

D.9 14-Nov-'22

Present:

- Yonatan
- Michael
- Yasin

Discussed:

- Front end and back end are being implemented. We are integrating today.
- Our approach has changed. We have recognized that the Model-View-ViewModel (MVVM) pattern was the best representation.
- Things to do:
  - Integrate front-end and back-end.
  - Revise databases.
  - Check on scope and extra features.
  - Testing and presentation notes to build.

D.10 24-Nov-'22

Present:

- Yonatan
- Michael

Not Present:

- Yasin

Discussed:

-
- Presentation is being worked on.
- Yonatan changed color scheme and logo.
- Things to do:
  1. He wants a better option for moods. A better option would be spinner or radio button. A perfect solution would be adding a slider [Seek bar].
  2. We need to focus more on accessibility.
  3. Add statistics.

D.11 27-Nov-'22

Present:

- Yonatan
- Michael

Not Present:

- Yasin

Discussed:

- Code must refactored to meet architectural standards.

- Code will be redone to the exact standard to the specified report
- Mood will be the focus.
- Medication and Symptoms are not the primary objective.
- Deadline to finish the code is tomorrow night.
- Deadline should be finished by Wednesday night.
- Deadline to finish the document components is Thursday.

D.12 28-Nov-'22

Present:

- Yonatan
- Michael

Not Present:

- Yasin

Discussed:

- Multiple meetings today.
- We have to go with MVC. MVVM was not feasible given the circumstances.
- Code is being redone to make the sections better.
- We want to get the code ready for deployment soon.
- Things to do:
  - Clean up redundant code
  - add comments
  - fix anything that seems broken.
  - Deploy on Docker.
  - Another addition of report documents and diagrams.

D.13 29-Nov-'22 until completion

Present:

- Yonatan
- Michael

Not Present:

- Yasin

Completed report, diagrams, and other loose ends.

Submitted on December 2nd, 2022.