Criterion C: Development

List of techniques used:

• Use of third-party libraries – Firebase Firestore

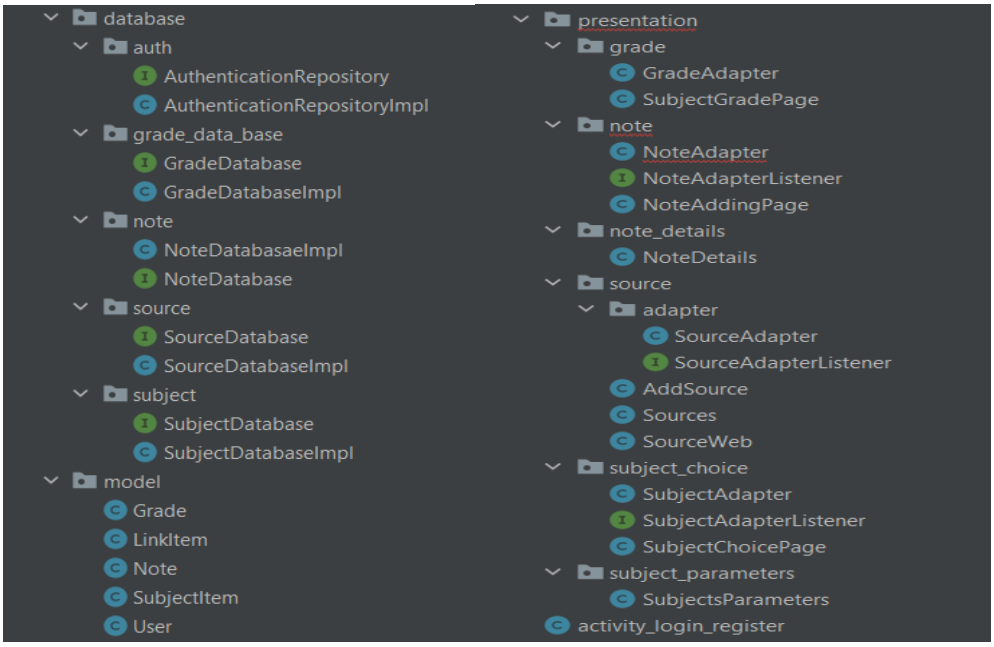
• File reading/writing – Firebase Firestore

• Algorithms – Database searching by field value

• Methods in classes – SubjectGradePage, NoteAddingPage, AddSource, Sources, SubjectChoicePage, SubjectsParameters, Activity_login_register.

• Database Implementations in classes – AuthenticationRepositoryImpl, GradeDatabseImpl, NoteDatabaseImpl, SourceDatabaseImpl, SubjectDatabaseImpl.

All the classes:



Java – Android Studio Code Breakdown:

The eBook helper application consists of many implementations and methods that lead to the customer's desired product. There are four main classes: Note, Grade and Source classes with their adding pages. The most complex codes are in these classes which will be discussed.

Retrieving information from Firebase Firestore:

The most important part of application is saving and displaying information from/through Firestore. The method of reading information is used in every "adding" pages. Therefore, I will provide the methods I used for reading information from the Firestore database as well as for Authentication.

```
2 usages
public class GradeDatabaseImpl implements GradeDatabase {
    1 usage
    private final FirebaseFirestore firestore = FirebaseFirestore.getInstance();
    3 usages
    private final CollectionReference collection = firestore.collection( collectionPath: "Grade");
    2 usages
    private final FirebaseAuth auth = FirebaseAuth.getInstance();
    1 usage
    @Override
    public Task<Void> createGrade(int grade, String subjectId) {
        String userId = auth.getCurrentUser().getUid();
        String gradeId = UUID.randomUUID().toString();
        Grade newGrade = new Grade(gradeId, subjectId, userId, grade);
        return collection.document(gradeId).set(newGrade);
    }

    2 usages
    @Override
    public Task<QuerySnapshot> getAllGrades(String subjectId) {
        String userId = auth.getCurrentUser().getUid();
        return collection.whereEqualTo( field: "ownerId", userId).whereEqualTo( field: "subjectId", subjectId).get();
    }

    @Override
    public Task<Void> deleteGrade(String gradeId) { return collection.document(gradeId).delete(); }
}
```

In this screenshot, Grade Database Implementation is given. Here, I implemented GradeDatabase class, in which I created Tasks (Voids) for grade, subjectId and gradeId. I need subjectId, so that subject parameters' Grade information will not mix together and each subject will display their own information with using their Id. Exactly in this class shown above, I gave Uids to user, grade and newGrade that will be added. Furthemore, "QuerySnapshot" is created in GradeDatabase class which is Firebase's task that allows me to gather all the information, in this case all the Grades. This is an example for Grade page function and it is all the same for Note and Source page, despite one Uid, where there are NoteId and SourceId.

```
1 usage
@Override
public void deleteNoteListener(Note note) {
    noteDatabase.deleteNote(note.noteId).addOnCompleteListener(task -> {
        if (task.isSuccessful()){
            getAllNote();
        }else {
            Toast.makeText( context: this, task.getException().getMessage(), Toast.LENGTH_SHORT).show();
        }
    });
}
```

When it come to delete function, It is available in Note as well as in Source class. This method calls noteDatabase.deletNote task which carried noteId and then with complete listener, AllNote function will be called where Id will be erased. In case if there will be any error, message will be toasted to user.

```
@Override
public void navigateSavedNote(Note note) {
    Intent intent = new Intent( packageContext: this, NoteDetails.class);
    SubjectItem subjectItem = (SubjectItem) getIntent().getSerializableExtra( name: "subject");
    intent.putExtra( name: "subject", subjectItem);
    intent.putExtra( name: "note", note);
    intent.putExtra( name: "isNoteSaved",  value: true);
    startActivity(intent);
}
}
```

Method shows in above picture, calls object Intent which is a good way to go from one page to another. Accordingly, NoteDetails.class is put, where user will be directed to the different page. Intent.putExtra just properly names new intent and carries subjectItem with is so the information will not mix.

```
binding.addNoteBtn2.setOnClickListener(view -> {
    AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
    View myView = LayoutInflater.from( context: this).inflate(R.layout.subject_layout, null);
    builder.setView(myView);
    EditText editText = myView.findViewById(R.id.subject_edit_text);
    builder.setTitle("Add subject");
    builder.setPositiveButton( text: "Add", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i) {
            String subjectName = editText.getText().toString();
            if (subjectName.isEmpty()) {
                Toast.makeText( context: SubjectChoicePage.this,  text: "Enter the subject name", Toast.LENGTH_SHORT).show();
            } else {
                subjectDatabase.addSubject(subjectName).addOnCompleteListener(task -> {
                    if (task.isSuccessful()) {
                        Toast.makeText( context: SubjectChoicePage.this,  text: "Successfully Added", Toast.LENGTH_SHORT).show();
                        getAllSubjects();
                    } else {
                        Toast.makeText( context: SubjectChoicePage.this, task.getException().getMessage(), Toast.LENGTH_SHORT).show();
                    }
                });
            }
        }
    });

    builder.create().show();
});
```

findViewById(R.Id) is replace with “binding”. This function is built in gradle and I use it, because it is more efficient and comfortable. Above, and example is shown where subject is added after clicking a button. LayoutInflater inflates the view from the layout and afterwards, title is set to “Add subject”. In onClick function, subjectName text is read by code and if it is empty, it will toast “Enter the subject name”, if it is filled with letters, subjectDatabase will be called where text will be saved and toast message will give shown to a user.

Same logic is used when I wrote code for SignIn and SignUp methods, but instead of subjectDatabase, AuthenticationDatabse is called where Firebase saved information in Authentication page:

Search by email address, phone number, or user UID

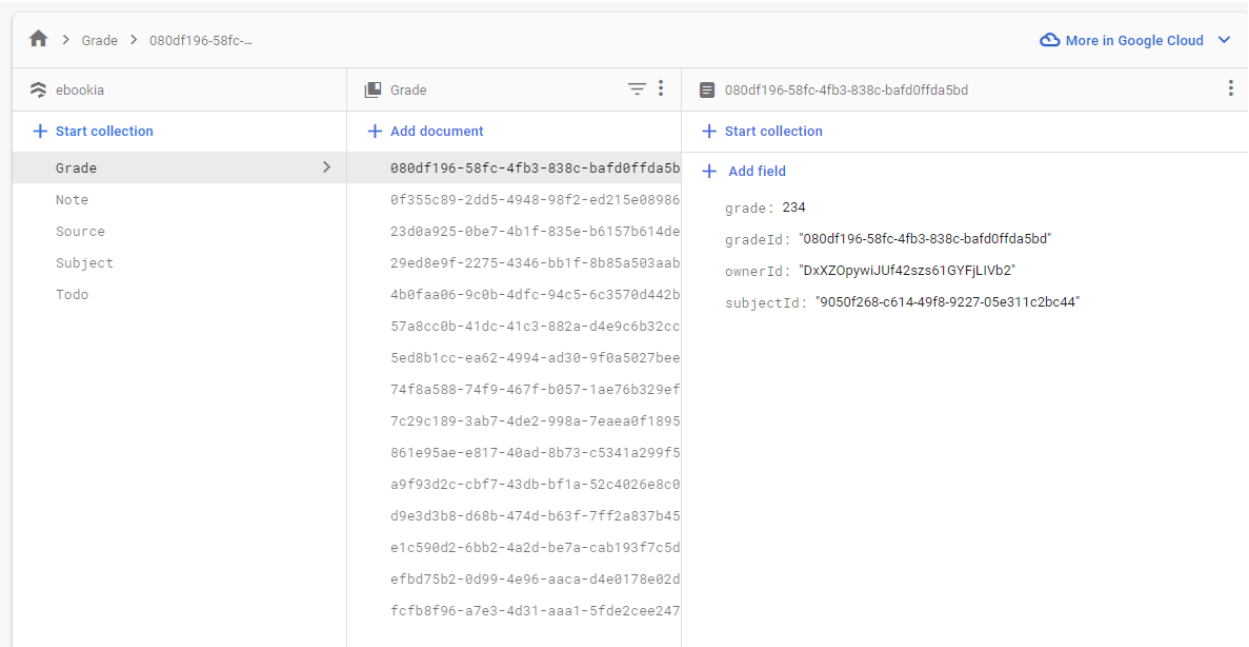
Add user

Identifier	Providers	Created	Signed In	User UID
gio.beridze33@gmail.com		Mar 10, 2023	Mar 10, 2023	YvmwJ07wRKPxh71jjBJ8cBpm3kr2
lukaggo2022@gmail.com		Feb 28, 2023	Feb 28, 2023	CXk8T1Pu2zNdiCG3XWgYQzecYo...
test@test.com		Feb 26, 2023	Feb 26, 2023	C9k1s8ktKFOO8hhncEBgoFx4fns1
1234567@gmail.com		Feb 26, 2023	Mar 10, 2023	DxXZOpywiJUf42szs61GYFjLIVb2
123456@gmail.com		Feb 26, 2023	Mar 10, 2023	HKqD30yCqvOIPOb0WIxOcQ2uW9...
12345@gmail.com		Feb 26, 2023	Feb 26, 2023	pYTuEw2LMnewnxJQZHXzwKEM...
gio2@gmail.com		Feb 26, 2023	Feb 26, 2023	NHp2gAyVNjTkjvPjFR7xVFg503e2
123123@gmail.com		Feb 26, 2023	Feb 26, 2023	SfGuVCea09hIfdm7TCUQLDMe47...
dfibjfdoiji@gmail.com		Feb 6, 2023	Feb 6, 2023	4s2NsBVqy2TFwDFaR6H6Irebi6w2
fd@gmail.com		Feb 4, 2023	Feb 4, 2023	HhxLFs4o3Xg16yh3kSJyhP00E2I2
gsjidj@gmail.com		Feb 3, 2023	Feb 3, 2023	XHeNCs68X7VevtPSASLdfqiD7Kw2
123@gmail.com		Feb 3, 2023	Feb 26, 2023	7yrBWQWtq2aiodjYCIHIASY11KF2
grkj@gmail.com		Feb 3, 2023	Feb 3, 2023	nUW9IbsonOO4AZ6IF7zfJO5ISx23
123fh@gmail.com		Feb 3, 2023	Feb 3, 2023	LOupdS6HvJfsDaHauQVWk0W3P...
jidghu@gmail.com		Feb 3, 2023	Feb 3, 2023	HMqQIkm1YqgqAtbHHfn8B2Kk5Ij1
ijasddf@gmail.com		Feb 3, 2023	Feb 3, 2023	NCWcyVRNUAPstcIGKaD5JSvDdg...

fd@gmail.com

Rows per page: 50 1 – 16 of 16

In Firebase firestore, shown below, every Grade, Note, Source and Subject have their own unique Id. By using that, database is free to recognize parameter, read it, save it and display it on the application.



Words: 531

Bibliography:

Firebase. “Add Data to Cloud Firestore | Firebase,” 2023. https://firebase.google.com/docs/firestore/manage-data/add-data#java_10.

Firebase. “Get Data with Cloud Firestore | Firebase,” 2023. https://firebase.google.com/docs/firestore/query-data/get-data.

Code, Bro. “Java Full Course for Free.” YouTube Video. *YouTube*, November 9, 2020. https://www.youtube.com/watch?v=xk4_1vDrzzo&t=41304s.