Design Document for Degrees

Group LM-302

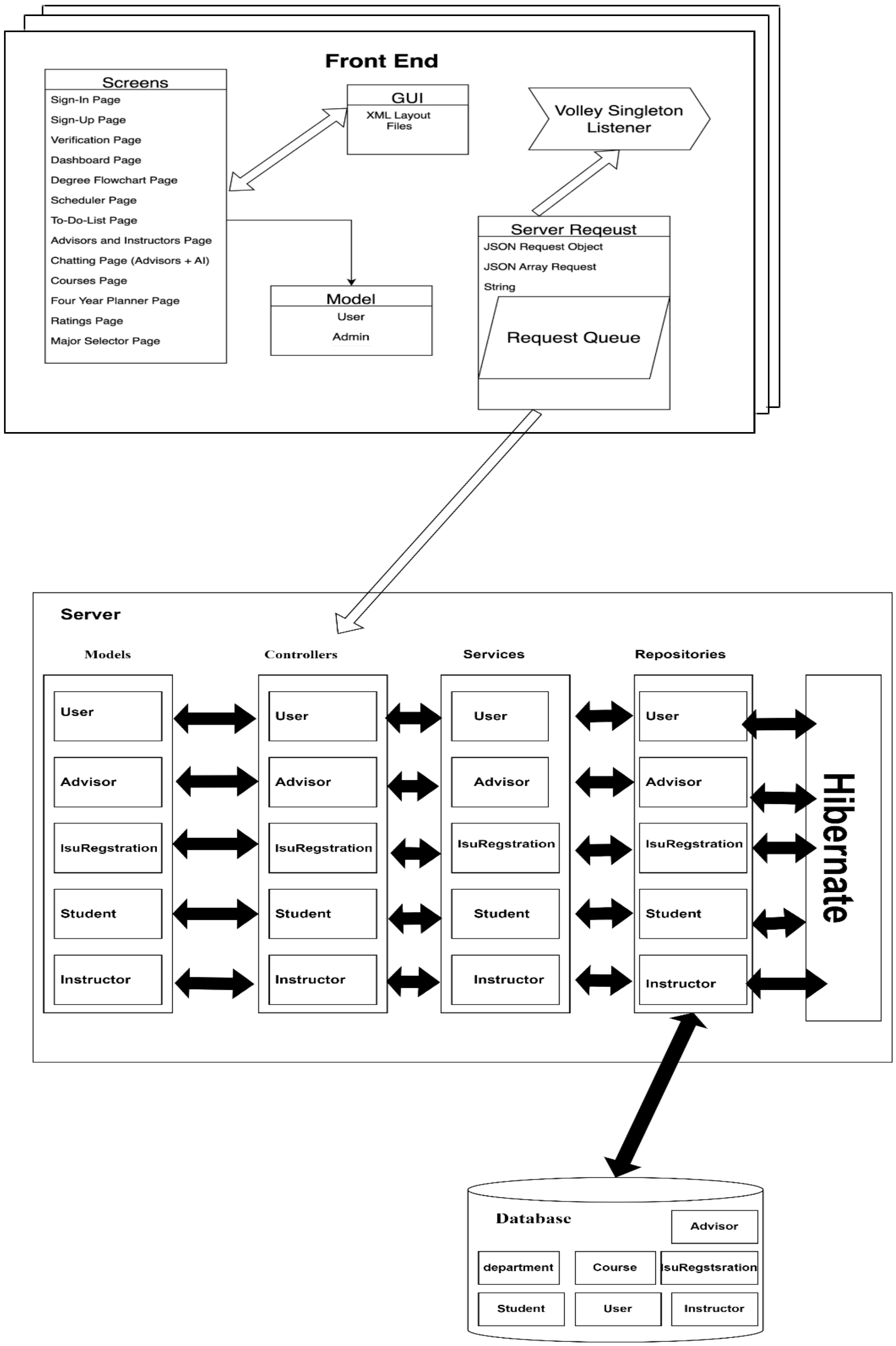
Mohammed: 25% contribution

Ellie: 25% contribution

Dev: 25% contribution

Blessing: 25% contribution

Block Diagram



Design Description

Frontend

The frontend is divided into Views, XML layout files and models.

Views

Views in Android represent the building blocks for user interface components in an Android application. These views can be configured and manipulated in your Android application's code to create the desired user interface and user experience.

These are some of the Interactive UI components that the Degrees app uses:

- TextView: Displays text to the user. It can be used to change the font, color, and text size, among other things. In the Degrees app, the Sign In title is a TextView.
- Button: An interactive button that responds to clicks. Buttons can have text, images, or both. In the Degrees app, the button uses texts mostly for example CreateNewUser and SignIn.
- EditText: A text entry widget that allows the user to type in data. For example, “email” and “Create New User” are EditText.
- ImageView: Displays an image to the user. It can load images from various sources like drawable resources, bitmaps, or URI for example the purple background color on the Sign in page.
- Spinner: A dropdown menu that allows the user to select one value from a list.
- ListView: A view group that displays a list of scrollable items shown on the Dashboard activity.

The views in the Degrees App are split into the following sections:

- Sign in screen, Verification code screen, Registration screen, Dashboard screen, Buttons and snippets for other screens, Degree flowchart screen, Courses screen and Chat screen.

XML files

XML (Extensible Markup Language) files play a crucial role in various aspects of software development especially Android Studio and data management. AndroidManifest.xml holds essential information about the Android application, such as its components (activities, services, broadcast receivers, and content providers), required permissions, and hardware and software features. XML is used to create drawable resources, like shapes, selectors, and layer-lists, providing a lightweight and scalable alternative to image files. XML files describe project dependencies, build processes, and plugins for example VolleySingleton.

Models

Models played a crucial role in structuring the Degrees App, promoting clean architecture and testable code. Models handle the logic for making network requests, processing responses, storing data, and dealing with errors. The model makes API calls to Postman and our actual server. They encompass the logic to create, read, update, and delete (CRUD) data, either from local databases, files, or network operations. In the Degrees app, the code shows several requests including GET and PUT requests (to change background color and theme) to the server.

Backend

The backend is divided into four layers: models, controllers, services, and repositories. Each layer is divided into objects. Each object within a layer represents a type of entity managed by the application (such as courses), or some other feature of the application (such as logging in).

Model layer

The model objects represent entities managed by the application. They are plain-old Java objects (POJOs), meaning that they are simple containers for data. They encapsulate all the information of a specific type of entity. For example, the User model stores the username field. They can either be detached or attached. Detached model objects represent entities created by the application’s business logic or sent by the client to the controller layer, and are not persisted to the database by default. Attached model objects represent entities retrieved from a repository, and changes to them are persisted to the database.

Controller layer

The controller objects implement the REST API of the application. They act as the entry point to the application, receiving requests from the client and sending back the corresponding responses. The controllers map each relevant REST API endpoint to a specific method that delegates the corresponding business logic to the service layer. Each endpoint represents a *resource*, such as an individual user or list of users, and can implement one or more of these HTTP methods:

- GET: The client requests to read the specified resource’s data.
- PUT: The client requests to overwrite the specified resource’s data with the data sent through the request body. Alternatively, the client requests to create a new resource at a specific location.
- POST: The client requests to add a new entry to the specified resource, without specifying a specific location for the new entry, such as adding a new user to a list of users.
- PATCH: The client requests to partially change the specified resource’s data, such as changing the email address of a user.
- DELETE: The client requests to delete the specified resource, such as deleting a course by ID.

Service layer

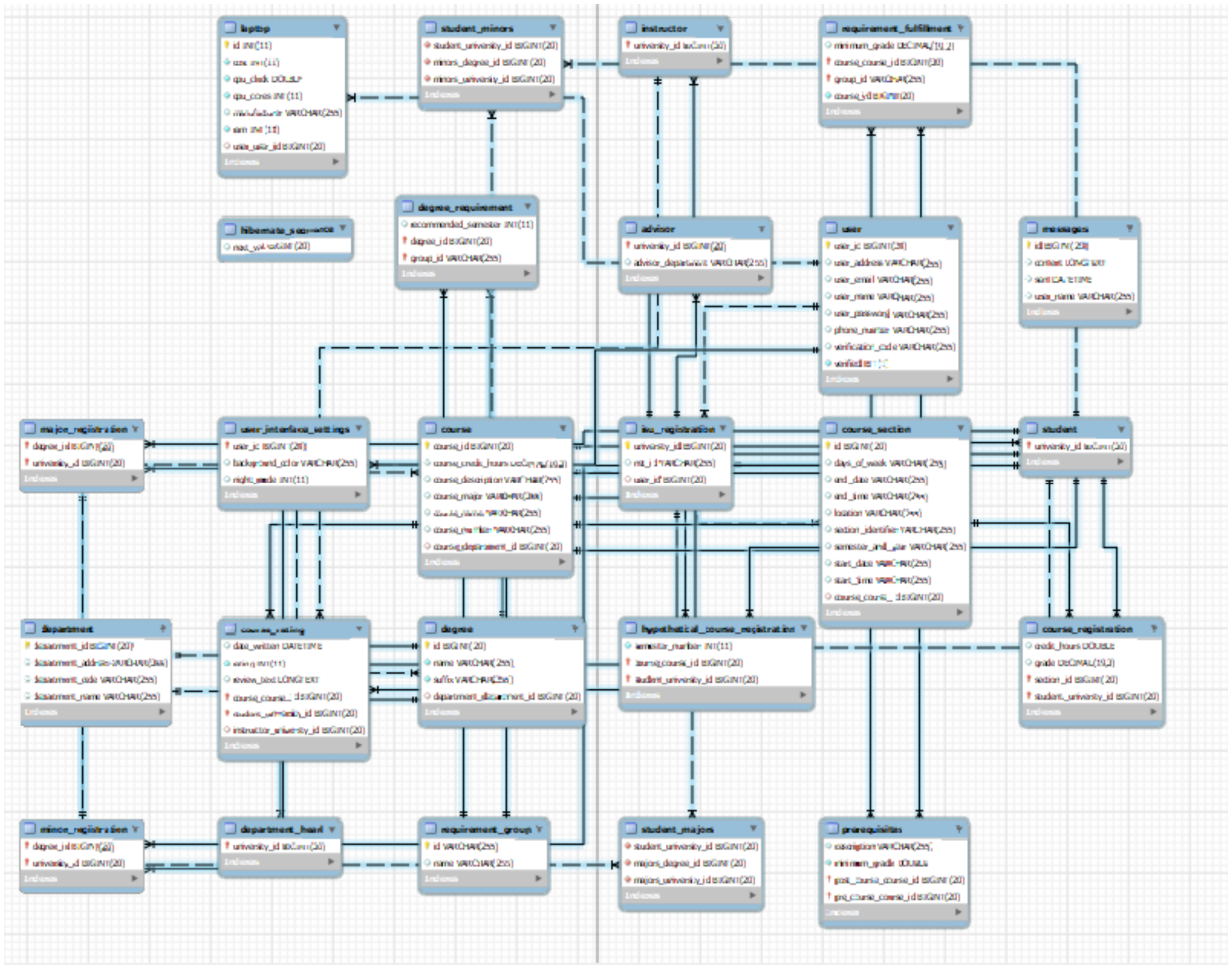
The service objects implement the application’s business logic. They expose abstract operations to the controller layer, such as getting a list of users. They delegate database access to the repository layer, and implement more complex application logic if needed, such as checking the validity of a login attempt.

Repository layer and database

The repository objects connect the service layer to the database using the Hibernate library. The underlying database, such as H2 or MySQL, is abstracted away. The repository objects instead expose a set of methods that represent CRUD operations, such as save, findById, etc. The repository objects are declared through interfaces, and the Spring Boot library automatically creates the implementations for them at runtime. The database stores data sent to it by the repository layer persistently. The data is retrieved later, sent to the upper layers in the backend, and eventually sent to the client.

Table Relationships

PUT THE TABLE RELATIONSHIPS DIAGRAM on this fourth page! (Create the picture using MySQLWorkbench)


laptop
id INT(11)
core INT(11)
cpu_clock DOUBLE
cpu_cores INT(11)
manufacturer VARCHAR(255)
ram INT(11)
user_user_id BIGINT(20)
student_minors
student_university_id BIGINT(20)
minors_degree_id BIGINT(20)
minors_university_id BIGINT(20)
instructor
university_id BIGINT(20)
requirement_fulfillment
minimum_grade DECIMAL(19,2)
course_course_id BIGINT(20)
group_id VARCHAR(255)
course_id BIGINT(20)
hibernate_sequence
next_val BIGINT(20)
degree_requirement
recommended_semester INT(11)
degree_id BIGINT(20)
group_id VARCHAR(255)
advisor
university_id BIGINT(20)
advisor_department VARCHAR(255)
user
user_id BIGINT(20)
user_address VARCHAR(255)
user_email VARCHAR(255)
user_name VARCHAR(255)
user_password VARCHAR(255)
phone_number VARCHAR(255)
verification_code VARCHAR(255)
verified BIT(1)
messages
id BIGINT(20)
content LONGTEXT
sent DATETIME
user_name VARCHAR(255)
major_registration
degree_id BIGINT(20)
university_id BIGINT(20)
user_interface_settings
user_id BIGINT(20)
background_color VARCHAR(255)
night_mode INT(11)
course
course_id BIGINT(20)
course_credit_hours DECIMAL(19,2)
course_description VARCHAR(255)
course_major VARCHAR(255)
course_name VARCHAR(255)
course_number VARCHAR(255)
course_department_id BIGINT(20)
isu_registration
university_id BIGINT(20)
net_id VARCHAR(255)
user_id BIGINT(20)
course_section
id BIGINT(20)
days_of_week VARCHAR(255)
end_date VARCHAR(255)
end_time VARCHAR(255)
location VARCHAR(255)
section_identifier VARCHAR(255)
semester_and_year VARCHAR(255)
start_date VARCHAR(255)
start_time VARCHAR(255)
course_course_id BIGINT(20)
student
university_id BIGINT(20)
department
department_id BIGINT(20)
department_address VARCHAR(255)
department_code VARCHAR(255)
department_name VARCHAR(255)
course_rating
date_written DATETIME
rating INT(11)
review_text LONGTEXT
course_course_id BIGINT(20)
student_university_id BIGINT(20)
instructor_university_id BIGINT(20)
degree
id BIGINT(20)
name VARCHAR(255)
suffix VARCHAR(255)
department_department_id BIGINT(20)
hypothetical_course_registration
semester_number INT(11)
course_course_id BIGINT(20)
student_university_id BIGINT(20)
course_registration
credit_hours DOUBLE
grade DECIMAL(19,2)
section_id BIGINT(20)
student_university_id BIGINT(20)
minor_registration
degree_id BIGINT(20)
university_id BIGINT(20)
department_head
university_id BIGINT(20)
requirement_group
id VARCHAR(255)
name VARCHAR(255)
student_majors
student_university_id BIGINT(20)
majors_degree_id BIGINT(20)
majors_university_id BIGINT(20)
prerequisites
description VARCHAR(255)
minimum_grade DOUBLE
post_course_course_id BIGINT(20)
pre_course_course_id BIGINT(20)
Indexes