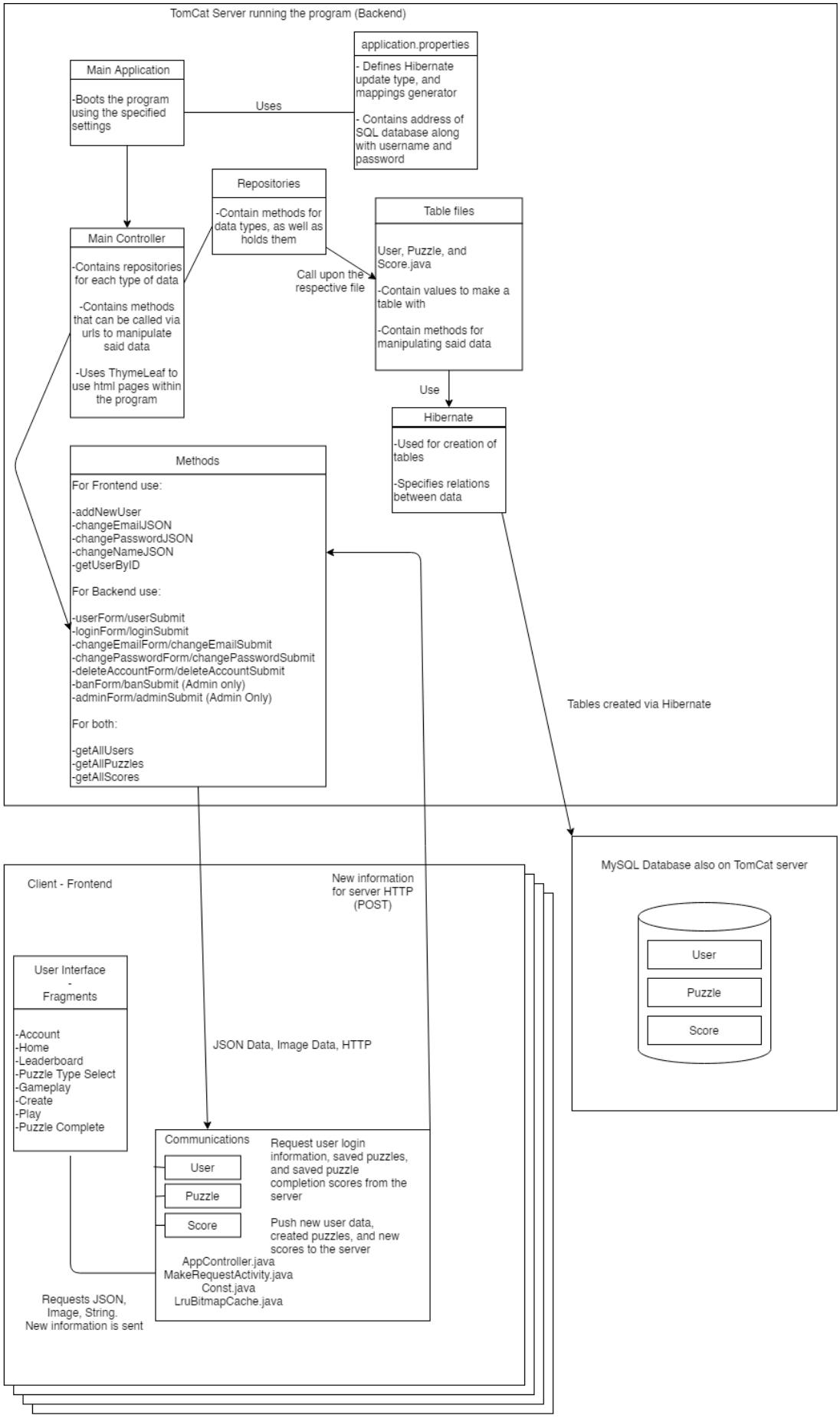
Design Document for Puzzle Game

By: Eric Kirch, Levi Jansen, Cris Medina, and Justin Le
Group mc_3


TomCat Server running the program (Backend)
application.properties
- Defines Hibernate update type, and mappings generator
- Contains address of SQL database along with username and password
Main Application
-Boots the program using the specified settings
Uses
Repositories
-Contain methods for data types, as well as holds them
Table files
User, Puzzle, and Score.java
-Contain values to make a table with
-Contain methods for manipulating said data
Call upon the respective file
Main Controller
-Contains repositories for each type of data
-Contains methods that can be called via urls to manipulate said data
-Uses ThymeLeaf to use html pages within the program
Use
Hibernate
-Used for creation of tables
-Specifies relations between data
Methods
For Frontend use:
-addNewUser
-changeEmailJSON
-changePasswordJSON
-changeNameJSON
-getUserByID
For Backend use:
-userForm/userSubmit
-loginForm/loginSubmit
-changeEmailForm/changeEmailSubmit
-changePasswordForm/changePasswordSubmit
-deleteAccountForm/deleteAccountSubmit
-banForm/banSubmit (Admin only)
-adminForm/adminSubmit (Admin Only)
For both:
-getAllUsers
-getAllPuzzles
-getAllScores
Tables created via Hibernate
MySQL Database also on TomCat server
User
Puzzle
Score
Client - Frontend
New information for server HTTP (POST)
User Interface - Fragments
-Account
-Home
-Leaderboard
-Puzzle Type Select
-Gameplay
-Create
-Play
-Puzzle Complete
JSON Data, Image Data, HTTP
Communications
User
Puzzle
Score
Request user login information, saved puzzles, and saved puzzle completion scores from the server
Push new user data, created puzzles, and new scores to the server
AppController.java
MakeRequestActivity.java
Const.java
LruBitmapCache.java
Requests JSON, Image, String. New information is sent

Design Descriptions:

Client:

The client side is the android graphical user interface which is what the user can expect to see when they boot up the app. All client-side files are held in our Android Studio project. There is a main activity which drives the app with all necessary fragments that will function as pages for the app. Activities are used to drive the functionality for each fragment and are used to help setup communication between the user interface and the server with respect to requests/responses. Each screen the user views has an xml file that defines what they see, a java file that initializes elements on the screen such as buttons, and, if necessary, an activity java file that handles communication between frontend and backend. These activity files are able to transfer user information as JSON objects and images stored from URLs. They are also able to call backend functions in order to change user information and store new information relating to scores for individual puzzles.

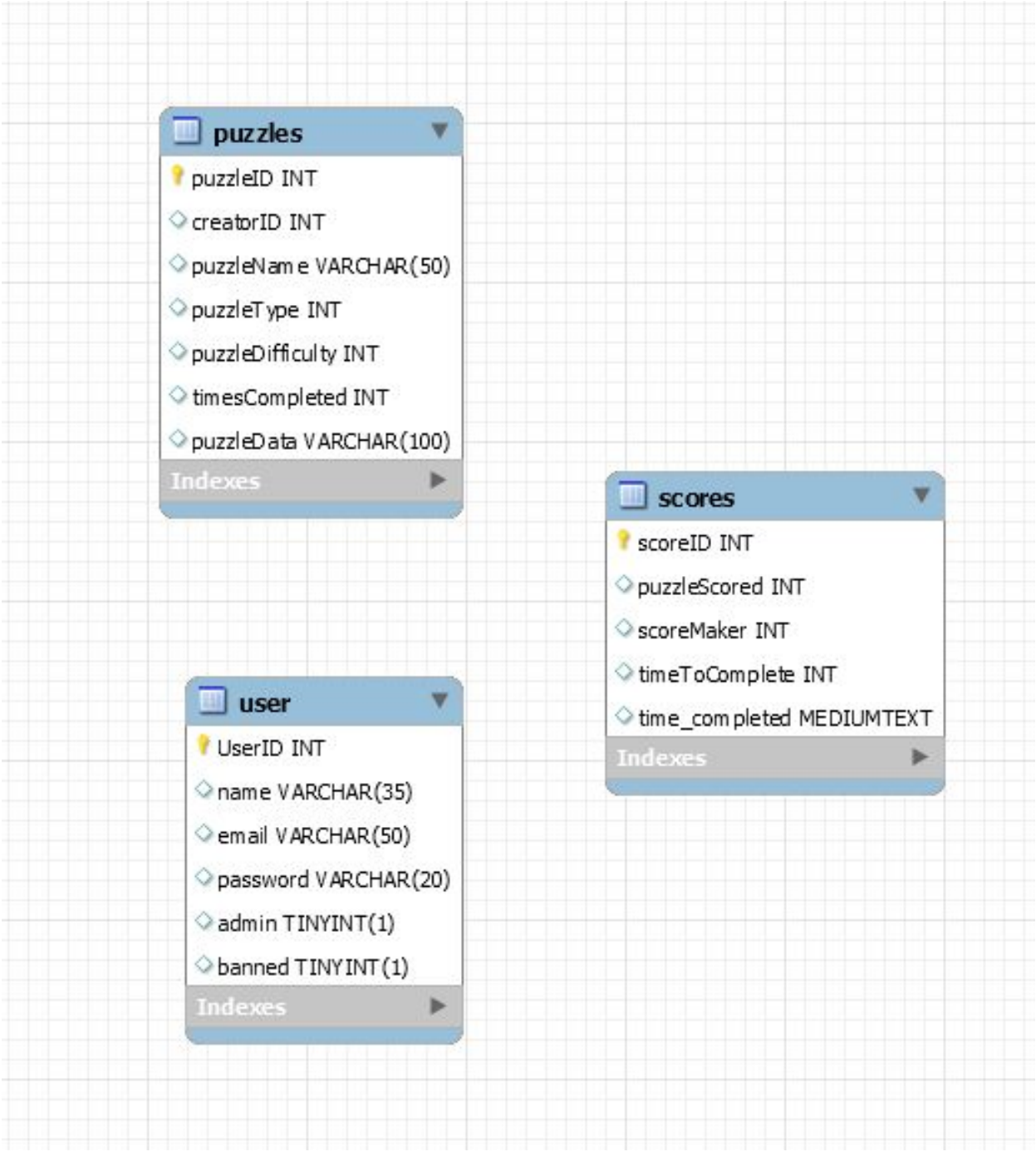
Server:

Our server contains all applications, methods, and controllers for handling information stored in the database. The server uses Hibernate to create new tables and relations between them. The three tables we currently have are users, puzzles, and scores. Users holds user login information, puzzles hold puzzle data, including the url to the puzzles image, while score holds its own id, as well as the user and puzzle of its respective maker and puzzle, along with the time it was completed, and time it took to complete. It handles connections to the database via JDBC configurations stored within the application.properties file. Images are stored in the home directory under userImages, and then a folder based off of the user's id.

Database:

Our application stores all information relevant to Users, Puzzles, and Scores in a MySQL database. Information for users includes a unique ID with which to recognize them, username, email address, password, whether they are an admin, and whether they are banned. Each puzzles is associated with a unique ID, the ID of the user that created the puzzle, a name, a type of puzzle (Jigsaw/Tile Sliding/Picross), a difficulty, the number of times it has been completed (for gauging popularity), and a url containing an image for the puzzle. Each score on a puzzle has a unique ID, the ID of the puzzle, the ID of the player who achieved the score, their time, and date of completion (for tiebreakers).

Below are the current tables that have been implemented into out program, while they are not final, they should give a good idea as to what we will use each table for



User

Variable Name	Data Type
UserID	Integer

name	String
email	String
password	String
admin	Boolean
banned	Boolean

Puzzle

Variable Name	Data Type
puzzleID	Integer
CreatorID	Integer
PuzzleName	String
PuzzleType	Integer
PuzzleDifficulty	Integer
TimesCompleted	Integer
PuzzleData	String

Score

Variable Name	Data Type
scoreID	Integer
PuzzleScored	Integer
ScoreMaker	Integer
TimeToComplete	Integer
TimeCompleted	Long