CS102	**Spring 2020/21**	Project Group G2F
Assistant:	**Mousa Farshkar Azari**	

~ Train Management System ~

Fliers

Alp Afyonluoğlu & Ali Emir Güzey

& Erkin Aydın & Cengizhan Terzioğlu

Criteria	TA/Grader	Instructor
Presentation		
Overall		

Project Detailed Design Report

(version 1.0)

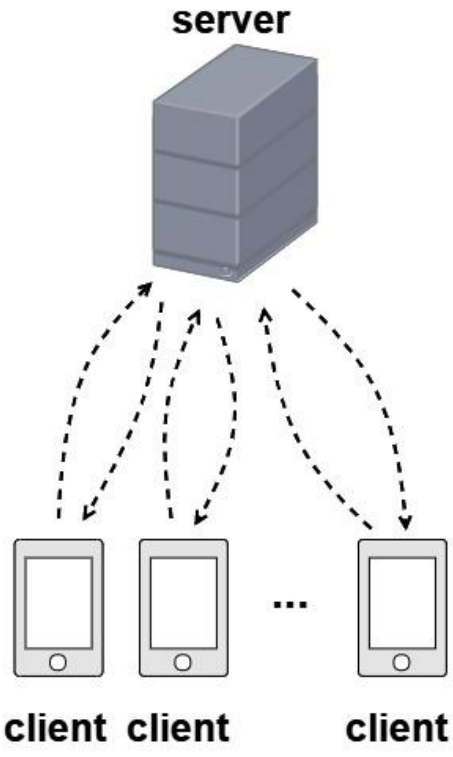
8 April 2021

1. Introduction

Our train management system, Trainly, will provide a common platform for both companies and customers, in which companies maintain their trains and customers easily find and buy tickets according to their advanced search preferences. The system will automate train movements and tasks given to employees by automatically assigning schedules that are uniquely created for each company according to their customer data.

2. Details

2.1 System Overview



2.1.1 Data Storage and Server-Client Communication

Android devices running the Trainly app will communicate with one another over a web server, which will be maintained with the use of Firebase platform. Realtime database of Firebase will be used as an online database. Client devices will communicate with the web server via web sockets and HTTPS requests, which will be handled by the Firebase API. In addition to saving required data to local storage of mobile devices, all user data will be stored in the remote database and, with the use of web sockets, client devices will be notified instantly when a change occurs in the database. In this way, some of the information, like statistics, will be updated in real time. For the client side, SQLite databases will be used to easily query data, like ticket data. SharedPreferences will be used on the Android side to store all data that is not stored in the SQLite database.

2.1.2 Application Development

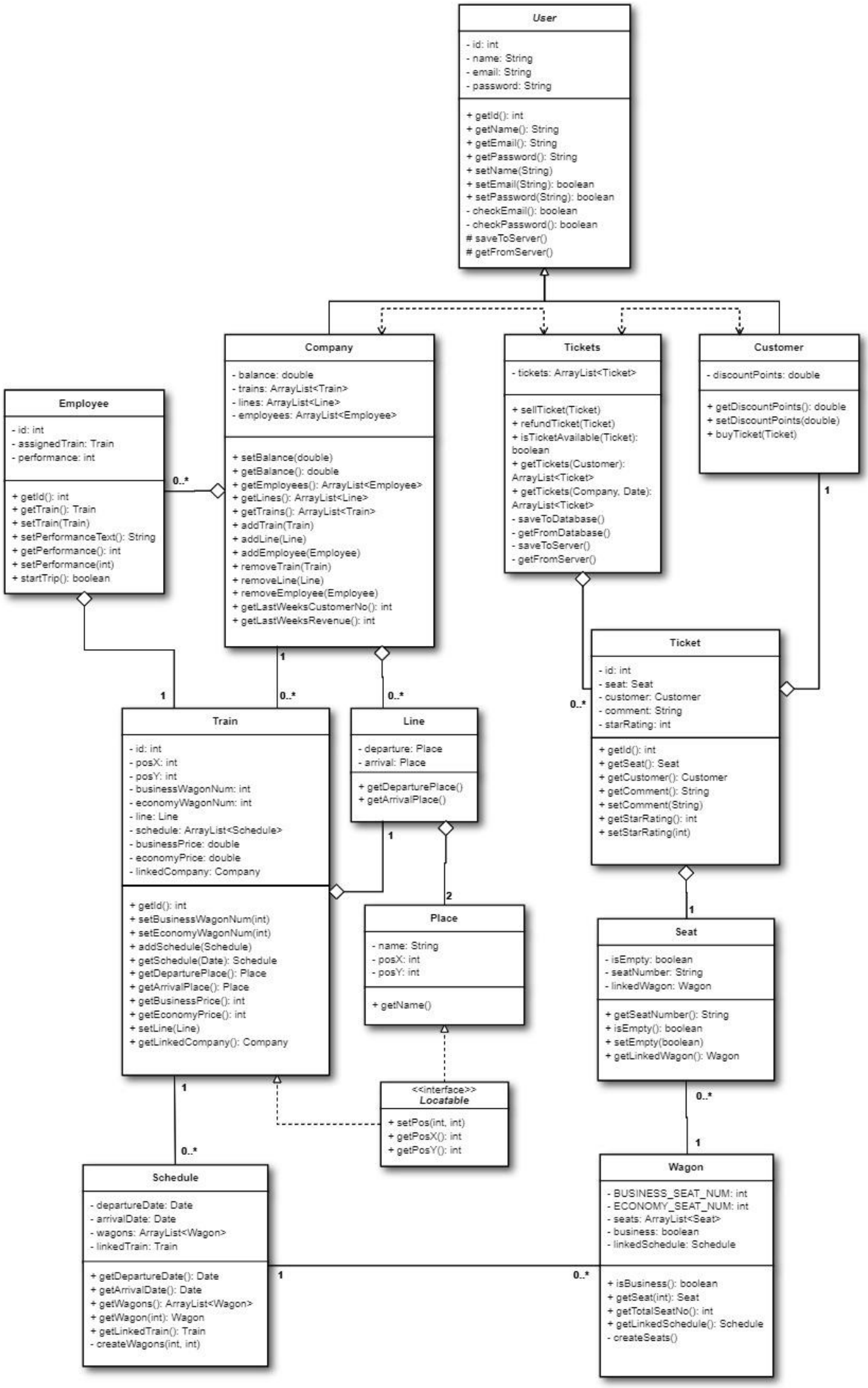
Android Studio will be used to develop the app, which will require the use of Java. XML files will be used for GUI development, which will include elements and layouts supported by Android. To be able to access, query and modify data taking place in the SQLite database, SQL commands will be used.

2.2 Core Design Details

2.2.1 Model Classes

This program consists of 12 classes and 1 interface. User class is an abstract class; Company and Customer classes extend it. Both User class and its subclasses have methods to save all user data to and receive data from the remote database. Company class represents the functions of a company client. This class has an aggregation relationship with the classes Employee and Line and a unidirectional association relationship with Train class. Employee class represents the machinists of Company; instances of this class start trips. Line class represents the train lines with departure and arrival places. It aggregates Place class to determine departure and arrival places. Train class sets schedules and implements the Locatable interface to be able to be tracked. Schedule class represents trips, creates wagons assigned to trips. Instances of Wagon class stores seats available for the trip in an array list. Seat class represents empty and occupied seats for a trip. Instances of Customer class choose a seat and buy the ticket object created for that seat.

Tickets class aggregates Ticket class and is used to put all tickets in an array list pool and return tickets that match with the expected query. It saves tickets to a local database and synchronizes this database with the remote database when required. Ticket class aggregates Customer and Seat classes. After its creation with given properties, Ticket class only has getter methods for Seat and Customer. Additionally, it has methods to modify and access comment and star rating of the ticket.


User
- id: int
- name: String
- email: String
- password: String
+ getId(): int
+ getName(): String
+ getEmail(): String
+ getPassword(): String
+ setName(String)
+ setEmail(String): boolean
+ setPassword(String): boolean
- checkEmail(): boolean
- checkPassword(): boolean
saveToServer()
getFromServer()
Company
- balance: double
- trains: ArrayList<Train>
- lines: ArrayList<Line>
- employees: ArrayList<Employee>
+ setBalance(double)
+ getBalance(): double
+ getEmployees(): ArrayList<Employee>
+ getLines(): ArrayList<Line>
+ getTrains(): ArrayList<Train>
+ addTrain(Train)
+ addLine(Line)
+ addEmployee(Employee)
+ removeTrain(Train)
+ removeLine(Line)
+ removeEmployee(Employee)
+ getLastWeeksCustomerNo(): int
+ getLastWeeksRevenue(): int
Tickets
- tickets: ArrayList<Ticket>
+ sellTicket(Ticket)
+ refundTicket(Ticket)
+ isTicketAvailable(Ticket): boolean
+ getTickets(Customer): ArrayList<Ticket>
+ getTickets(Company, Date): ArrayList<Ticket>
- saveToDatabase()
- getFromDatabase()
- saveToServer()
- getFromServer()
Customer
- discountPoints: double
+ getDiscountPoints(): double
+ setDiscountPoints(double)
+ buyTicket(Ticket)
Employee
- id: int
- assignedTrain: Train
- performance: int
+ getId(): int
+ getTrain(): Train
+ setTrain(Train)
+ setPerformanceText(): String
+ getPerformance(): int
+ setPerformance(int)
+ startTrip(): boolean
Ticket
- id: int
- seat: Seat
- customer: Customer
- comment: String
- starRating: int
+ getId(): int
+ getSeat(): Seat
+ getCustomer(): Customer
+ getComment(): String
+ setComment(String)
+ getStarRating(): int
+ setStarRating(int)
Train
- id: int
- posX: int
- posY: int
- businessWagonNum: int
- economyWagonNum: int
- line: Line
- schedule: ArrayList<Schedule>
- businessPrice: double
- economyPrice: double
- linkedCompany: Company
+ getId(): int
+ setBusinessWagonNum(int)
+ setEconomyWagonNum(int)
+ addSchedule(Schedule)
+ getSchedule(Date): Schedule
+ getDeparturePlace(): Place
+ getArrivalPlace(): Place
+ getBusinessPrice(): int
+ getEconomyPrice(): int
+ setLine(Line)
+ getLinkedCompany(): Company
Line
- departure: Place
- arrival: Place
+ getDeparturePlace()
+ getArrivalPlace()
Place
- name: String
- posX: int
- posY: int
+ getName()
Seat
- isEmpty: boolean
- seatNumber: String
- linkedWagon: Wagon
+ getSeatNumber(): String
+ isEmpty(): boolean
+ setEmpty(boolean)
+ getLinkedWagon(): Wagon
<<interface>>
Locatable
+ setPos(int, int)
+ getPosX(): int
+ getPosY(): int
Schedule
- departureDate: Date
- arrivalDate: Date
- wagons: ArrayList<Wagon>
- linkedTrain: Train
+ getDepartureDate(): Date
+ getArrivalDate(): Date
+ getWagons(): ArrayList<Wagon>
+ getWagon(int): Wagon
+ getLinkedTrain(): Train
- createWagons(int, int)
Wagon
- BUSINESS_SEAT_NUM: int
- ECONOMY_SEAT_NUM: int
- seats: ArrayList<Seat>
- business: boolean
- linkedSchedule: Schedule
+ isBusiness(): boolean
+ getSeat(int): Seat
+ getTotalSeatNo(): int
+ getLinkedSchedule(): Schedule
- createSeats()
0..*
1
2

2.3 Task Assignment

Ali Emir Güzey*	Company, Employee, Line, Place
Erkin Aydın*	Train, Wagon, Seat
Alp Afyonluoğlu	User, Tickets, Locatable, XML Files
Cengizhan Terzioğlu*	Customer, Ticket, Schedule

*GUI related control classes will be divided to these members

3. Summary & Conclusions

Further information about class relations and details of the project are provided in this report. After determining required classes, task assignment and separation between group members are made.