Chapter 3
System Analysis and Design

3.1 Aim of chapter:-

In this chapter team represent main functions of “My School”.

3.2 System analysis:-

What is System analysis?

It is a process of collecting and interpreting facts, identifying the problems, and decomposition of a system into its components.

Purpose of System Analysis:-

System analysis is conducted for the purpose of studying a system or its parts in order to identify its objectives. It is a problem solving technique that improves the system and ensures that all the components of the system work efficiently to accomplish their purpose. Analysis specifies what the system should do.

3.2.1 Requirement gathering techniques:-

Requirement gathering techniques are techniques that describe how tasks are performed under specific circumstances. A task may have none or one or more related techniques. A technique should be related to at least one task.

The following are techniques we used to gather requirements for My School:-

1- Brainstorming:-

Brainstorming is used in requirement gathering to get as many ideas as possible from group of people. Generally used to identify possible solutions to problems, and clarify details of opportunities.

2- Previous Projects’ Analysis:-

Reviewing the websites and mobile applications of existing systems to create basic system specifications, we even reviewed the requirements that drove creation of the existing system – a starting point for documenting current requirements. Important pieces of information are often buried in existing systems that help us ask questions as part of validating requirement completeness.

3- Interview:-

Interviews of schools and users are critical to creating the software. Without understanding the goals and expectations of the users and schools, we are very unlikely to satisfy them. We also have to recognize the perspective of each interviewee, so that, we can properly weigh and address their inputs.

4- Observation:-

By observing users, the analyst identify the process flow, steps, pain points and opportunities for improvement. Observations can be passive or active (asking questions while observing). Passive observation is better for getting feedback on a prototype (to refine requirements), where active observation is more effective at getting an understanding of an existing business process.

5- Survey/Questionnaire:-

When collecting information from many schools too many to interview with budget and time constraints a survey or questionnaire is used. The survey can force schools to select from choices, rate something (“Agree Strongly, agree…”), or have open ended questions allowing free-form responses.

3.2.2 Project requirements

Project requirements are conditions or tasks that must be completed to ensure the success or completion of the project. They provide a clear picture of the work that needs to be done. They consist of two types functional and non-functional requirements.

Functional Requirements

User requirements:-

1– User:-

- The user shall be able to create a user account to be able to apply for a school.
- The user shall be able to search for schools either in a list or on a map view.
- The user shall be able to filter his/her school search by city, area, type, fee range and interests.
- The user shall be able to view a school profile to view biography, location on map, pictures, interests and levels' details.
- The user shall be able to fill a dynamically generated application form from each school to apply.
- The user shall view his/her profile to check on requests' progress.
- The user shall receive an application code for each appointed request to present for the school.

2– School administrator:-

- The school administrator shall be able to register his/her school to appear in search results.
- The school administrator shall be able to edit the school profile.
- The school administrator shall be able to keep track the account's transactions.
- The school administrator shall be able to create new school programs
- The school administrator shall be able to edit details of each level in a school program.
- The school administrator shall be able to generate his/her own school application
- The school administrator shall be able to open and close the advising.
- The school administrator shall be able to make an appointment for requests he/she receive from the user.
- The school administrator shall be able to reject a request from user.
- The school administrator shall be able to view filled applications from attended users.
- The school administrator shall be able to view statistical charts.

3– System administrator:-

- The system administrator shall be able to view sign-up requests from schools.
- The system administrator shall be able to set city and area for a school.
- The system administrator shall be able to set longitude and latitude of a school.
- The system administrator shall be able to accept or reject school sign-up.
- The system administrator shall be able to view statistical charts.

System requirements:-

- On making a request for an application from a school. The requestor shall be presented with a dynamic application form that records details of the student that wants to apply.
- On making a request for a search query for schools. The user shall be presented with the query result in 0.50 seconds or less.
- All school application forms must be indexed by schoolID, UserID and by the LevelID.
- My School shall maintain a log of all transactions that have been made to the system.

Non-Functional Requirements

1– Usability:-
The user interface for My School shall be implemented as simple bootstrap to support ease of use to the user.

2– Safety:-
The system shall not disclose any information from a filled application to the school before the appointment is attended.

3– Privacy:-
The system shall not disclose any information from a filled application to the operators of the system.

4– Portability:-
The system shall work on desktop pc, laptop, smartphones and tablets anywhere at anytime as long as there is internet connection.

3.2.3 Flow Chart:-

What is Flow chart?
A flowchart is a type of diagram that represents an algorithm, workflow or process. Flowchart can also be defined as a diagrammatic representation of an algorithm (step by step approach to solve a task).

Purpose of Flow chart:-
Flowcharts are used in designing and documenting simple processes or programs. Like other types of diagrams, they help visualize what is going on and thereby help understand a process, and perhaps also find less-obvious features within the process, like flaws and bottlenecks.

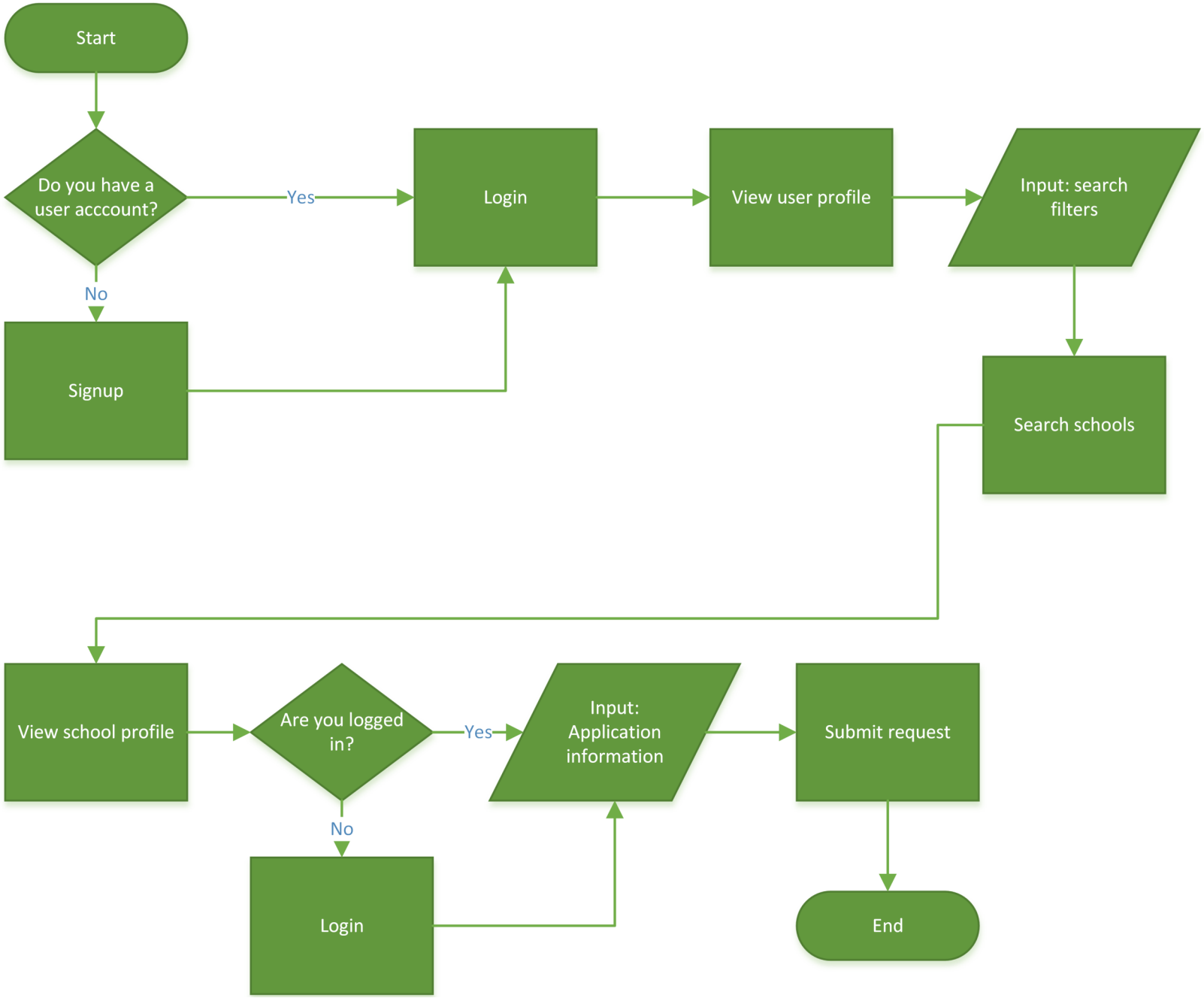


Figure 3.1—User's Flow Chart

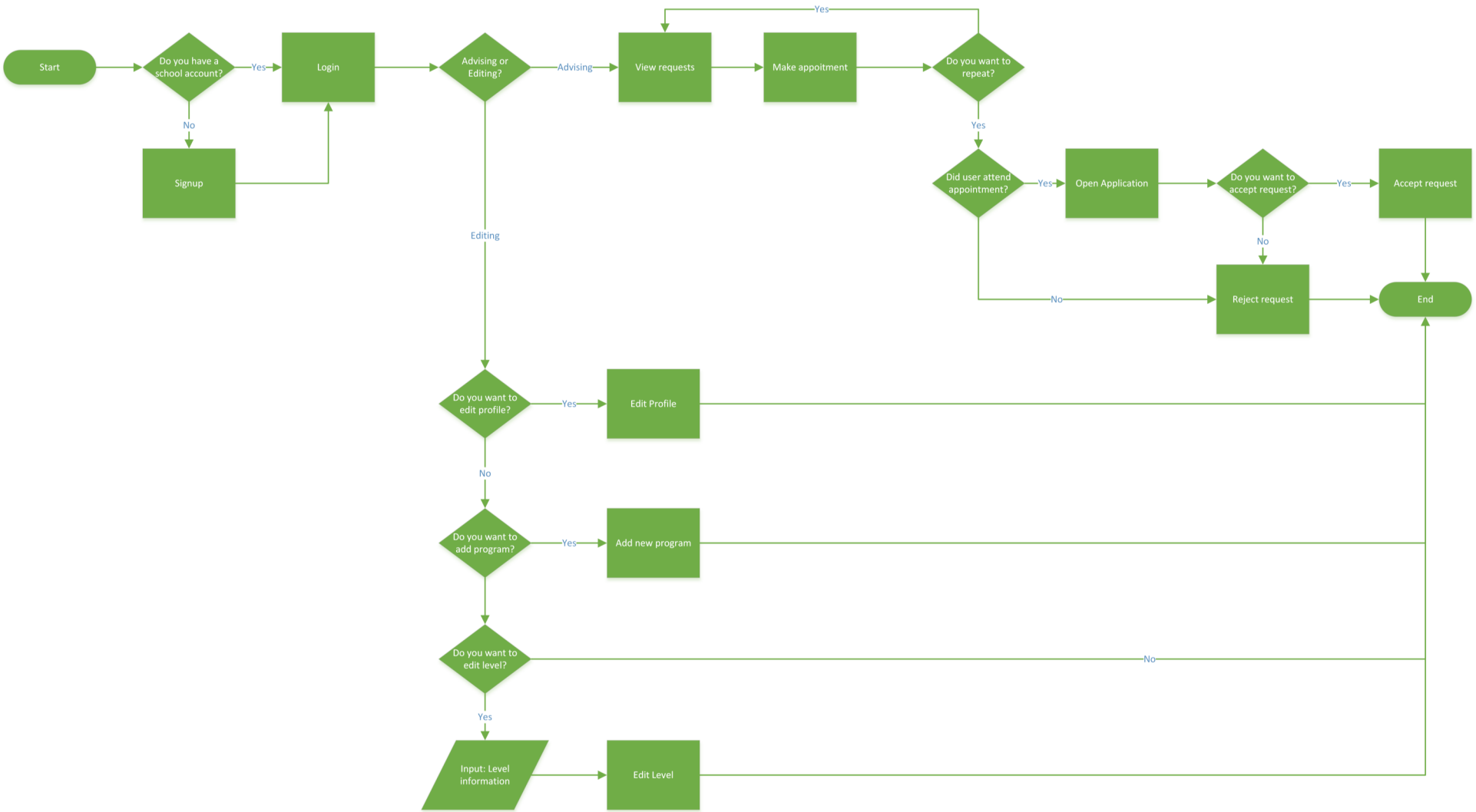


Figure 3.2—School's Flow Chart

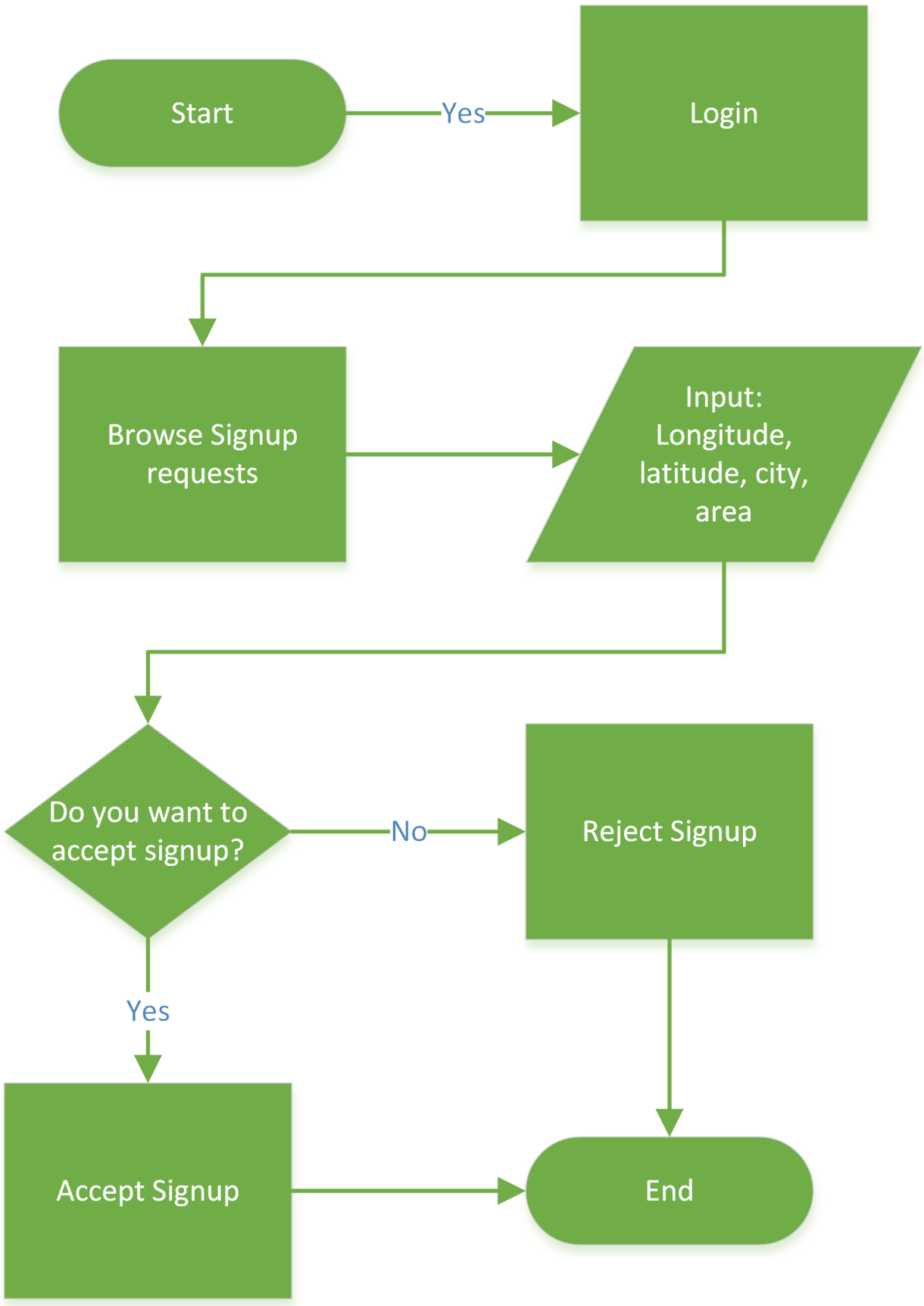


Figure 3.3—Admin's Flow Chart

3.2.4 Use Case:-

What is Use Case?

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses.

Purpose of Use Case:-

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analyzed to gather its functionalities, use cases are prepared and actors are identified.

In brief, the purposes of use case diagrams can be said to be as follows:-

- Used to gather the requirements of a system.
- Used to get an outside view of a system.
- Identify the external and internal factors influencing the system.
- Show the interaction among the requirements are actors.

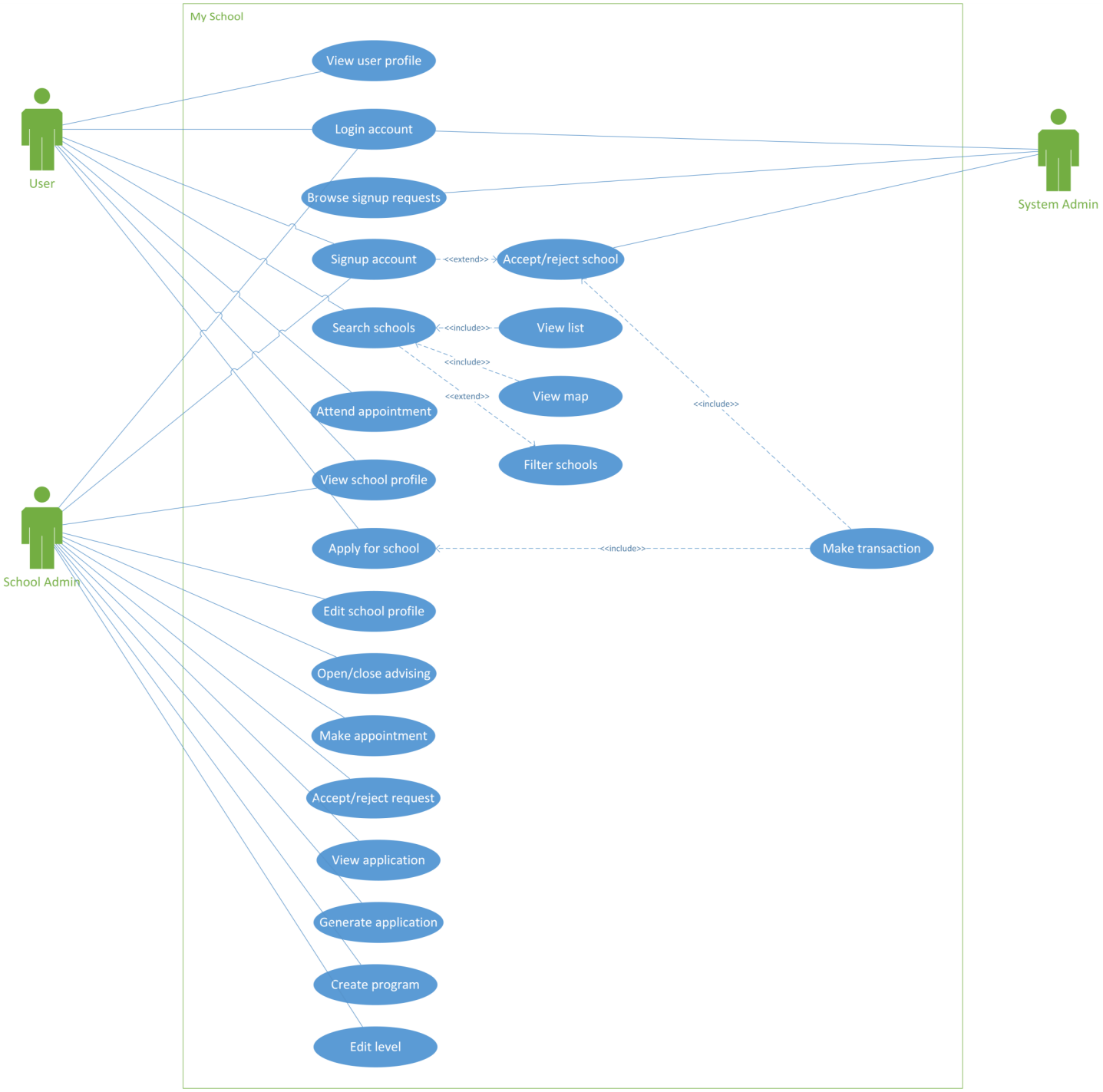


Figure 3.4—My School's Use case diagram

3.2.5 Class Diagram:-

What is Class Diagram?

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

In the diagram, classes are represented with boxes that contain three compartments:-

- The top compartment contains the name of the class. It is printed in bold and centered, and the first letter is capitalized.
- The middle compartment contains the attributes of the class. They are left-aligned and the first letter is lowercase.
- The bottom compartment contains the operations the class can execute. They are also left-aligned and the first letter is lowercase.

Purpose of Class Diagram:-

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the structure of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

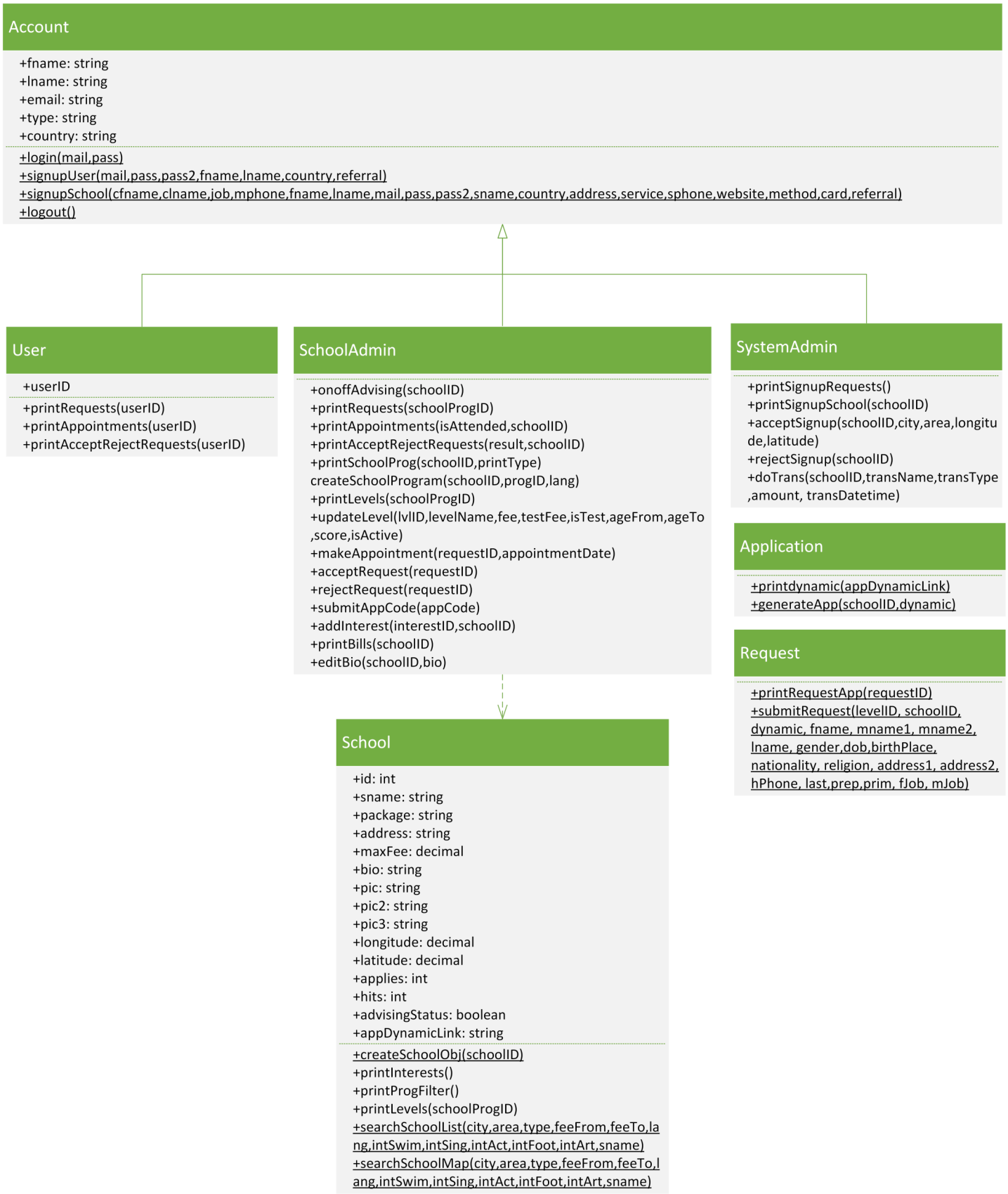


Figure 3.5—My School's Class diagram

3.2.6 Sequence Diagram:-

What is Sequence Diagram?

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

Purpose of Sequence Diagram:-

- Model high-level interaction between active objects in a system
- Model the interaction between object instances within a collaboration that realizes a use case
- Model the interaction between objects within a collaboration that realizes an operation
- Either model generic interactions (showing all possible paths through the interaction) or specific instances of a interaction (showing just one path through the interaction)

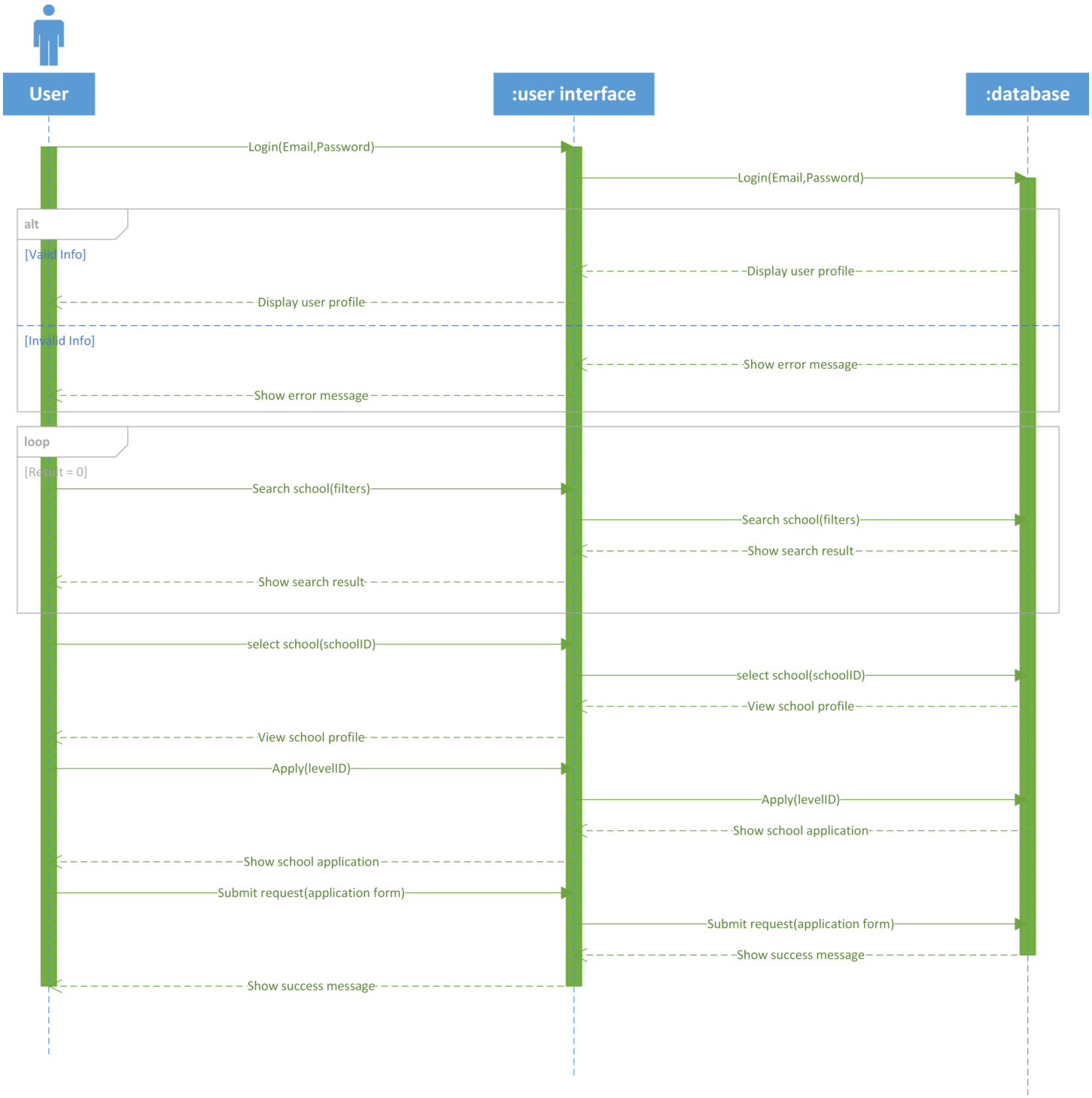


Figure 3.6—Sequence diagram (User)

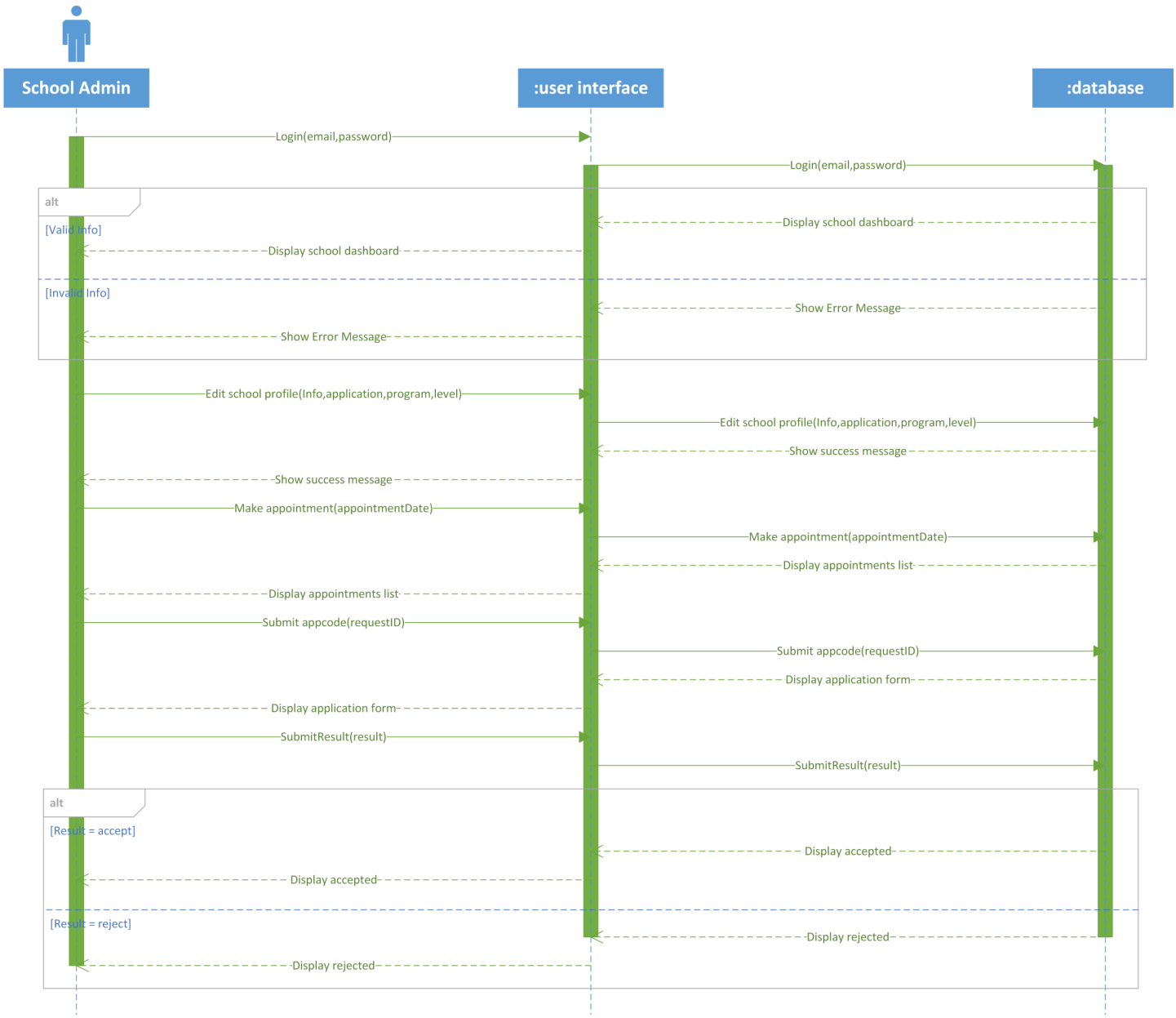


Figure 3.7—Sequence diagram (School)

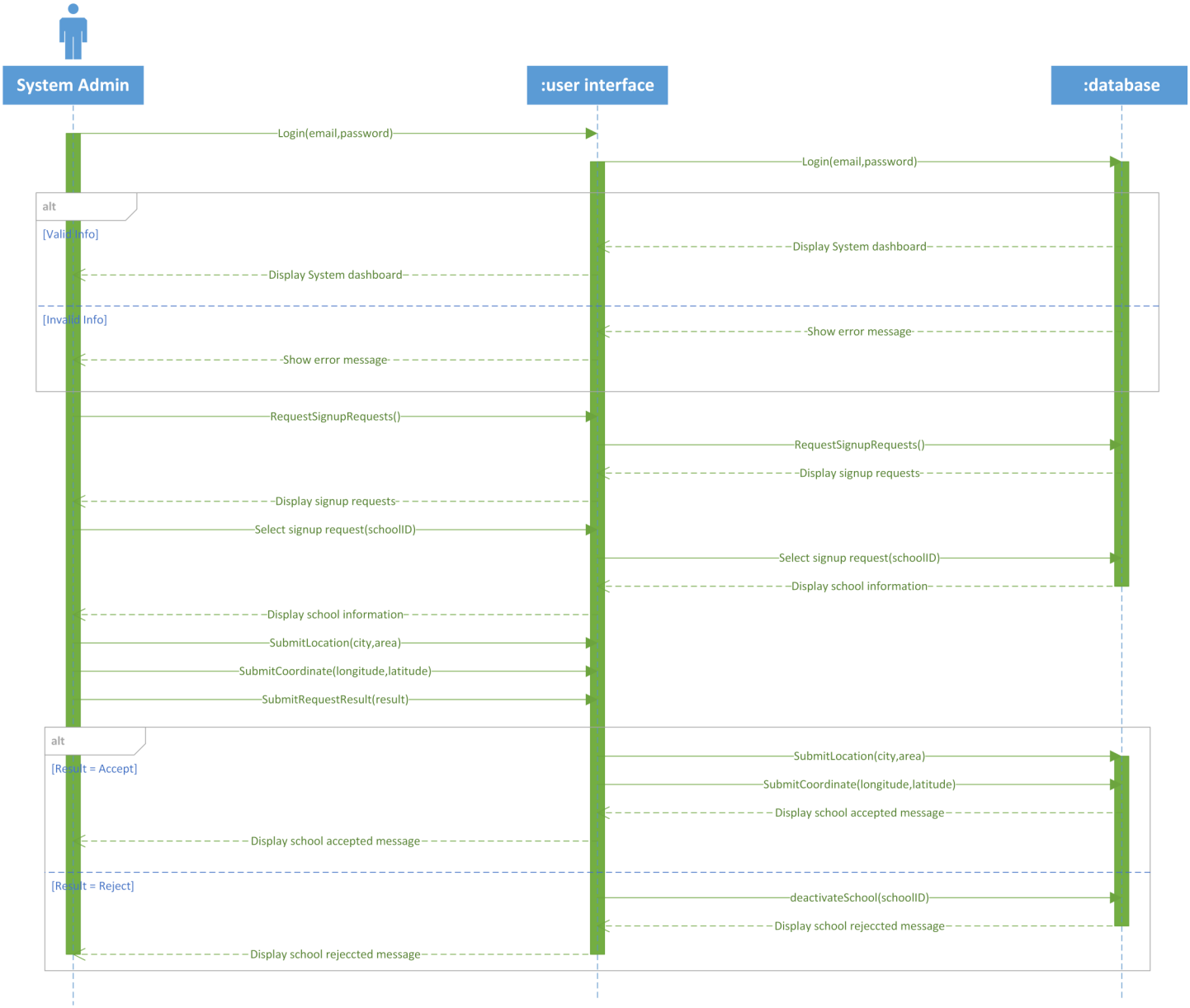


Figure 3.8—Sequence diagram (System)

3.3 System design:-

What is System design?

It is a process of planning a new business system or replacing an existing system by defining its components or modules to satisfy the specific requirements. Before planning, you need to understand the old system thoroughly and determine how computers can best be used in order to operate efficiently.
System Design focuses on how to accomplish the objective of the system.

Purpose of System design:-

Systems design's purpose is to define, develop and design the architecture, modules, interfaces, and data for a system to satisfy specified requirements.

3.3.1 Entity Relationship Diagram:-

What is ERD?

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is a component of data. In other words, ER diagrams illustrate the logical structure of databases. At first glance an entity relationship diagram looks very much like a flowchart It is the specialized symbols, and the meanings of those symbols, that make it unique.

Components of ERD:-

Entities	Which are represented by rectangles. An entity is an object or concept about which you want to store information.
Relationship	How entities act upon each other or are associated with each other. Think of relationships as verbs. For example, the named student might register for a course. The two entities would be the student and the course, and the relationship depicted is the act of enrolling, connecting the two entities in that way. Relationships are typically shown as diamonds or labels directly on the connecting lines.
Attribute	A property or characteristic of an entity. Often shown as an oval or circle.
Cardinality	Show whether the relationship is 1-1, 1-many or many-to-many

ERD Methodology:-

1. Identify Entities	Identify the roles, events, locations, tangible things or concepts about which the end-users want to store data.
2. Find Relationships	Find the natural associations between pairs of entities using a relationship matrix.
3. Draw Rough ERD	Put entities in rectangles and relationships on line segments connecting the entities.
4. Fill in Cardinality	Determine the number of occurrences of one entity for a single occurrence of the related entity.
5. Define Primary Keys	Identify the data attribute(s) that uniquely identify one and only one occurrence of each entity.
6. Draw Key-Based ERD	Eliminate Many-to-Many relationships and include primary and foreign keys in each entity.
7. Identify Attributes	Name the information details (fields) which are essential to the system under development.
8. Map Attributes	For each attribute, match it with exactly one entity that it describes.
9. Draw fully attributed ERD	Adjust the ERD from step 6 to account for entities or relationships discovered in step 8.
10. Check Results	Does the final Entity Relationship Diagram accurately depict the system data?

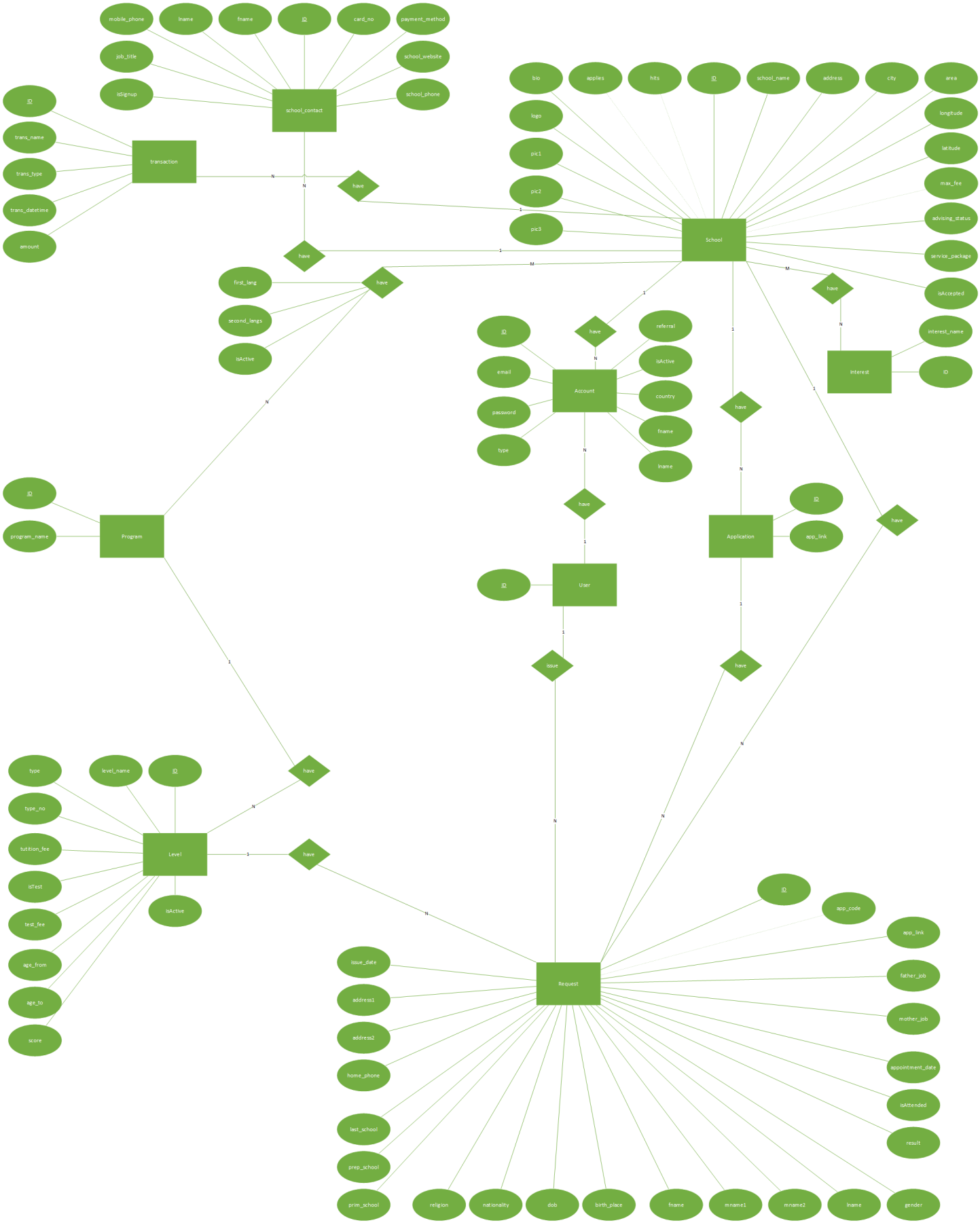


Figure 3.9—My School's ERD

3.3.2 Database Schema:-

What is Database Schema?

A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.

A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It's the database designers who design the schema to help programmers understand the database and make it useful.

Database Instance:-

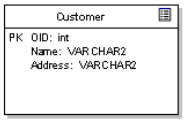
It is important that we distinguish these two terms individually. Database schema is the skeleton of database. It is designed when the database doesn't exist at all. Once the database is operational, it is very difficult to make any changes to it. A database schema does not contain any data or information.

A database instance is a state of operational database with data at any given time. It contains a snapshot of the database. Database instances tend to change with time. A DBMS ensures that its every instance (state) is in a valid state, by diligently following all the validations, constraints, and conditions that the database designers have imposed.

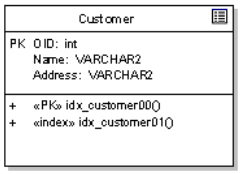
The UML Data Model Profile:-

The Data Model Profile is a UML extension that supports the modeling of relational databases in UML. It includes custom extensions for such things as tables, data base schema, table keys, triggers, and constraints. While this is not a ratified extension, it still illustrates one possible technique for modeling a relational database in the UML.

Components of UML Data Model Profile:-



Tables & Columns A table in the UML Data Profile is a class with the «Table» stereotype, displayed as above with a table icon in the top right corner. Database columns are modeled as attributes of the «Table» class.



Behavior Describe the behavior associated with columns, including indexes, keys, triggers, procedures, and so on. Behavior is represented as stereotyped operations.

Relationships The UML data modeling profile defines a relationship as a dependency of any kind between two tables. It is represented as a stereotyped association and includes a set of primary and foreign keys.

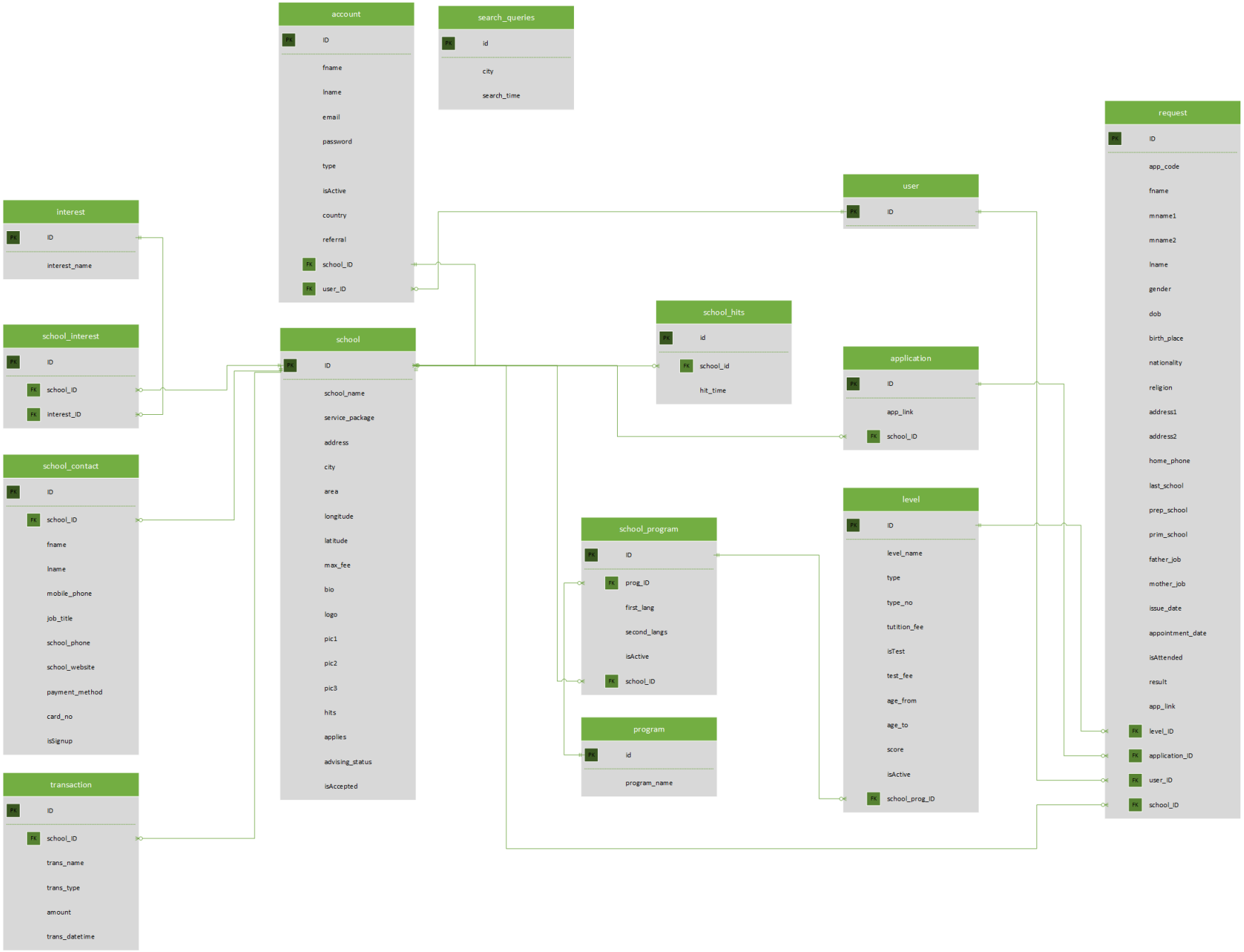


Figure 3.10—My School's Schema