

System Design and Architecture for DropIN

Contributors:

Denis Dekhtyarenko
Iangola Andrianarison
Jesse Yao
Justin Ding
Shaahid Khan
Sharon Xiao

Table of Contents

1.	CRC Cards.....	3
1.1.	Users.....	3
1.2.	Group.....	3
1.3.	Activity.....	4
1.4.	Chat.....	4
1.5.	DAO.....	5
2.	Software Architecture Diagram.....	6
3.	System Decomposition.....	7

CRC Cards

Class Name: User	
Parent: None	
Subclass: None	
Responsibilities: <ul style="list-style-type: none"> • Knows Username • Knows Password • Knows Email Address • Knows Location • Knows followed activities • Knows the friend list • Able to Login • Able to Logout • Able to Delete • Able to create account 	Collaborators: <ul style="list-style-type: none"> • Chat • Group

Class Name: Group	
Parent: None	
Subclass: None	
Responsibilities: <ul style="list-style-type: none"> • Knows all the users in the groups • Knows the type of activities • Knows the skill level for the group • Knows location of the group, if any 	Collaborators: <ul style="list-style-type: none"> • Chat • Users • Activity

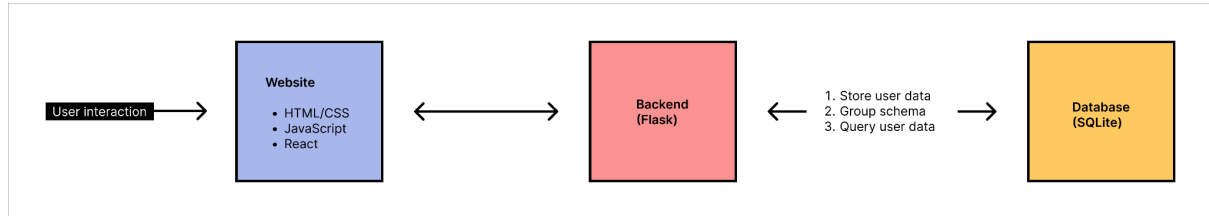
Class Name: Activity	
Parent: None	
Subclass: None	
Responsibilities: <ul style="list-style-type: none"> • Knows the type of activity • Knows the list of all of the available activities • Knows the list of all the groups engaged in that activity • Knows all the users following that activity 	Collaborators: <ul style="list-style-type: none"> •

Class Name: Chat	
Parent: None	
Subclass: None	
Responsibilities: <ul style="list-style-type: none"> • Knows all the users in the groups • Knows the number of users • Knows the time and date • knows the contents of the message • Sends the message containing the content • Knows the user who sends the message • Knows the user/users who receive the message 	Collaborators: <ul style="list-style-type: none"> • Users • Activity • Group

Class Name: DAO	
Parent: None	
Subclass: None	
Responsibilities: <ul style="list-style-type: none"> • Register user based on username, password and email • Login users using their email and password • Modifies account information • verifies account information • Deletes the account • Store users' profile picture • Create groups with group name, associated activity, skill level, group picture and a generated group ID • Send and retrieve chat messages associated with a group ID • Modifies group information • Store and retrieve activities • Search for groups and specific group information • Get list of activities • Get user account information • Get user activity preferences (Skill level and location) • Get user friend list • Add/delete from user friend list • Create user contact requests 	Collaborators: <ul style="list-style-type: none"> • Chat • Users • Activity • Group

Software Architecture Diagram

Three-tiered architecture link: <https://www.linuxjournal.com/article/3508>



System Decomposition

Presentation Layer:

Our front-end is composed of HTML, CSS, and JavaScript (using React libraries). This is where our user will be interacting with our application.

Application Layer:

Our backend uses Python3, NodeJS, and Flask and is connected to a MySQL database. When sending data between the front-end and back-end of our system, HTTP endpoints are used in the back-end. On the front-end, HTTP POST requests allow us to create and update classes through the application layer. Information is retrieved via GET requests from the database.

Data Layer:

Our back-end interacts with the MySQL database to make changes to the data and perform read/write operations.

Error Handling:

If the user inputs invalid information, they will receive an error message informing them of their error (e.g. incorrect login info, search query that yields no results). In the case of a network or external system failure, our strategy is to return the user back to their dashboard/profile page once the network is restored so that they can resume their activity.