StudyBuddies
SYSTEM DESIGN DOCUMENT

ASHWIN MAYURATHAN
BRANDAN BURGESS
CHINMAY GOKHALE
DAMIAN MELIA

Contents

Frontend

Excluded class name and subclasses for react components

React Component(s): HomeScreen	
Responsibilities: - Render the view GoogleMaps current location - View major buildings on campus	Collaborators: ● React (useState, useEffect, Text, View, StyleSheet) ● react-router-dom (useParams) ● MapView(PROVIDER_Google)

React Component(s): AccountPage	
Responsibilities: - Render the view for user information and settings - Provide text forms for updating user data - Provide forms for updating user courses	Collaborators: - React (useState, useEffect, Text, View, StyleSheet) - react-router-dom (useParams)

React Component(s): NavBar	
Responsibilities:	

- Render the view for switching screens on the app	Collaborators: - React (useState, useEffect, Text, View, StyleSheet) - react-router-dom (useParams) - GoogleMapsScreen - AccountPage

React Component(s): SwipeUpMenu	
Responsibilities: - Menu for students to find study spots - View capacity, occupancy and courses studied in major locations on campus	Collaborators: ● React (useState, useEffect, Text, View, StyleSheet) ● react-native-swipe-up-down ●

React Component(s): StudyTimer	
Responsibilities: - Customizable pomodoro timer for students studying while using app	Collaborators: ● React (useState, useEffect, Text, View, StyleSheet) ● React-countdown ●

React Component(s): GoalsBar	
Responsibilities: - Menu for students to create goals during the semester	Collaborators: ● React (useState, useEffect, Text, View, StyleSheet) ● react-native-swipe-up-down ●

React Component(s): Groups	
Responsibilities: - Page for students to form groups	Collaborators: ● React (useState, useEffect, Text, View, StyleSheet)

React Component(s): StudyPage	
Responsibilities: - Join Courses, view analytics about study times and course grades, and timer to track/log study times/habits	Collaborators: ● React (useState, useEffect, Text, View, StyleSheet) ● React-countdown ●

React Component(s): UserProfile	
Responsibilities: - View name, bio and courses of user	Collaborators: ● React (useState, useEffect, Text, View, StyleSheet) ●

React Component(s): FriendProfile

Responsibilities:	Collaborators:
- View name, bio and courses of user	● React (useState, useEffect, Text, View, StyleSheet) ●

Backend

Classes/Structs

Class Name: DefaultModel	
Interfaces: None	
Responsibilities: - Defines the basic `gorm.model` struct: - ID: uint - Created At: timestamp - Updated At: timestamp -	Collaborators: - None

Class Name: User	
Interfaces: UserService, UserDataStore	
Responsibilities: - Simply defines the structure of the `User` data type - DefaultModel - Auth0ID string - Username string - Avatar string - Name string	Collaborators: - DefaultModel

- Courses []Course - Friends []User - Courses []Course - StudyLog map[string]uint - UserCourses map[string]map[string]uint	

Class Name: Course	
Interfaces: UserService, UserDataStore	
Responsibilities: - Simply defines the structure and interfaces of the `Course` data type - DefaultModel - Name string - Image string - NumStudents int - Students []User	Collaborators: - Defaultmodel

Class Name: Message	
Interfaces: MessageService, MessageDataStore	
Responsibilities: - Simply defines the structure of a Message - Content *string - SenderId string - ChatId string	Collaborators: - Defaultmodel

Class Name: Chat
Interfaces: ChatService, ChatDataStore

Responsibilities: - Simply defines the structure of a Chat - Name string - Messages []Message - Users []User - OwnerID uint - LastEvent time.Time	Collaborators: - Defaultmodel

Class Name: Building	
Interfaces: BuildingService, BuildingDataStore	
Responsibilities: - Simply defines the structure of a Building - Name string - Image string - BuildingCode string - Rooms []Room	Collaborators: - Defaultmodel

Class Name: Room	
Interfaces: RoomService, RoomDataStore	
Responsibilities: - Simply defines the structure of a Room - RoomNumber string - Image string - Courses []Course - BuildingCode uint - Capacity uint - Occupancy uint - Students []User	Collaborators: - Defaultmodel

Interfaces

src/server/service/user_service.go

Interface Name: UserService	
Responsibilities: - Returns The Information gathered from the \`data\` layer Functions: - Register(user *User) (*User, error) - GetUser(id string) (*User, error) - DeleteUser(id string) error - GetCourses(id string) ([]Course, error) - JoinCourse(userID string, courseName string) error - LeaveCourse(userID string, courseName string) error - AddFriend(userID, friendID string) error - GetFriendByUsername(username string) (*User, error) - RemoveFriend(userID, friendID string) error - GetFriends(userID string) ([]User, error)	Collaborators: - User - Course

src/server/datastore/user_datastore.go

Interface Name: UserDataStore	
Responsibilities:	Collaborators:

- All operations involving users within the DB - User CRUD operations - Course joining/leaving - CreateUser(user *User) (*User, error) - GetUserByID(id string) (*User, error) - DeleteUser(id string) error - GetCourses(id string) ([]Course, error) - JoinCourse(userID string, courseName string) error - LeaveCourse(userID string, courseName string) error - AddFriend(userID, friendID string) error - GetFriendByUsername(username string) (*User, error) - RemoveFriend(userID, friendID string) error - GetFriends(userID string) ([]User, error)	- User - Course - Errors - Gorm - strings

src/server/service/course_service.go

Interface Name: CourseService	
Responsibilities: - Returns The Information gathered from the `data` layer Functions: - CreateCourse(course *Course) (*Course, error) - GetCourse(name string) (*Course, error) - DeleteCourse(name string) error - GetAllCourses() ([]Course,	Collaborators: - User - Course

error) - GetStudents(name string) ([]User, error) - AddStudent(courseName string, studentID string) error - RemoveStudent(courseName string, studentID string) error	

src/server/datastore/course_datastore.go

Interface Name: CourseDataStore	
Responsibilities: - All operations involving courses within the DB - Course CRUD operations - CreateCourse(course *Course) (*Course, error) - GetCourseByName(name string) (*Course, error) - DeleteCourse(name string) error - GetAllCourses() ([]Course, error) - GetStudents(name string) ([]User, error) - AddStudent(courseName string, studentID string) error - RemoveStudent(courseName string, studentID string) error	Collaborators: - User - Course - Errors - Gorm

Interface Name: MessageService

Responsibilities: - Intermediate Layer between the Handler and DB layer - CreateMessage(message *Message) (*Message, error) - GetMessages(chat *Chat) (*[]MessageWithUser, error) - UpdateMessage(message *Message) error - DeleteMessage(message *Message) error	Collaborators: - Message

Interface Name: MessageDatastore	
Responsibilities: - All DB CRUD operations regarding messages - CreateMessage(message *Message) (*Message, error) - GetMessagesFromChat(chat *Chat) (*[]MessageWithUser, error) - UpdateMessage(message *Message) error - DeleteMessage(message *Message) error	Collaborators: - Message

Interface Name: ChatService	
Responsibilities: - Intermediate layer between the Handler and DB layers - CreateChat(chat *Chat) (*Chat, error) - GetChat(ID string) (*Chat, error) - UpdateChat(chat *Chat) (*Chat, error) - DeleteChat(ID string) error - GetAllChats(userID string) ([]Chat, error) - GetUsers(ID string) ([]User, error) - AddUser(ID, userID string) error - RemoveUser(ID, userID string) error	Collaborators: - Chat

Interface Name: ChatDatastore	
Responsibilities: - All DB CRUD operations regarding chats - CreateChat(chat *Chat) (*Chat, error) - GetChatByID(ID string) (*Chat, error) - UpdateChat(chat *Chat) (*Chat, error) - DeleteChat(ID string) error	Collaborators: - Chat

- GetAllChats(userID string) ([]Chat, error) - GetUsers(ID string) ([]User, error) - AddUser(ID, userID string) error - RemoveUser(ID, userID string) error	

Interface Name: BuildingService	
Responsibilities: - Intermediate layer between Handler and DB layers - CreateBuilding(building *Building) (*Building, error) - GetBuilding(code string) (*Building, error) - DeleteBuilding(code string) error - GetRooms(code string) ([]Room, error) - AddRoom(buildingCode string, roomNumber string) error - RemoveRoom(buildingCode string, roomNumber string) error	Collaborators: - Building

Interface Name: BuildingDatastore

Responsibilities:	Collaborators:
- All DB CRUD operations regarding buildings	- Building
- CreateBuilding(building *Building) (*Building, error)	
- GetBuilding(code string) (*Building, error)	
- DeleteBuilding(code string) error	
- GetRooms(code string) ([]Room, error)	
- AddRoom(buildingCode string, roomNumber string) error	
- RemoveRoom(buildingCode string, roomNumber string) error	

Interface Name: RoomService	
Responsibilities:	Collaborators:
- Intermediate layer between Handler and DB layers	- Room
- CreateRoom(room *Room) (*Room, error)	
- GetRoom(number string, buildingCode string) (*Room, error)	
- DeleteRoom(number string, buildingCode string) error	
- GetCourses(number string, buildingCode string) ([]Course, error)	
- AddCourse(roomNumber	

string, buildingCode string, courseName string) error - RemoveCourse(roomNumber string, buildingCode string, courseName string) error - SetCapacity(roomNumber string, buildingCode string, capacity int) error - SetOccupancy(roomNumber string, buildingCode string, occupancy int) error - AddOccupant(roomNumber string, buildingCode string, studentID string) error - RemoveOccupant(roomNumber string, buildingCode string, studentID string) error - GetOccupants(roomNumber string, buildingCode string) ([]User, error)	

Interface Name: RoomDatastore	
Responsibilities: - All DB CRUD operations regarding rooms - CreateRoom(room *Room) (*Room, error) - GetRoom(number string, buildingCode string) (*Room, error) - DeleteRoom(number	Collaborators: - Room

string, buildingCode string) error - GetCourses(number string, buildingCode string) ([]Course, error) - AddCourse(roomNumber string, buildingCode string, courseName string) error - RemoveCourse(roomNumber string, buildingCode string, courseName string) error - SetCapacity(roomNumber string, buildingCode string, capacity int) error - SetOccupancy(roomNumber string, buildingCode string, occupancy int) error - AddOccupant(roomNumber string, buildingCode string, studentID string) error - RemoveOccupant(roomNumbe r string, buildingCode string, studentID string) error - GetOccupants(roomNumber string, buildingCode string) ([]User, error)	

Handlers

src/server/handlers/handler.go

Handler Name: Handler

Responsibilities:	Collaborators:
- Syncing the handlers with a \`gin router\` to be passed to the server - Creates the route groups: /api - /account - /register - /auth/callback - /login - /delete - /courses - /join - /leave - /course - /create - /delete - /students - /add_student - /del_student	- User - Course - Jwt-go - Gin - net/http - encoding/json - Fmt - UserService - CourseService

src/server/handlers/user_handler.go

Handler Name: UserHandler	
Responsibilities: - Handling all requests regarding the \`User\` struct and returning the corresponding responses back to the client - AuthCallback (handles Auth0 token management) - Register - Login - Delete - GetCourses - JoinCourse - LeaveCourse	Collaborators: - User - Course - Jwt-go - Gin - net/http - encoding/json - Fmt - UserService - CourseService

src/server/handlers/course_handler.go

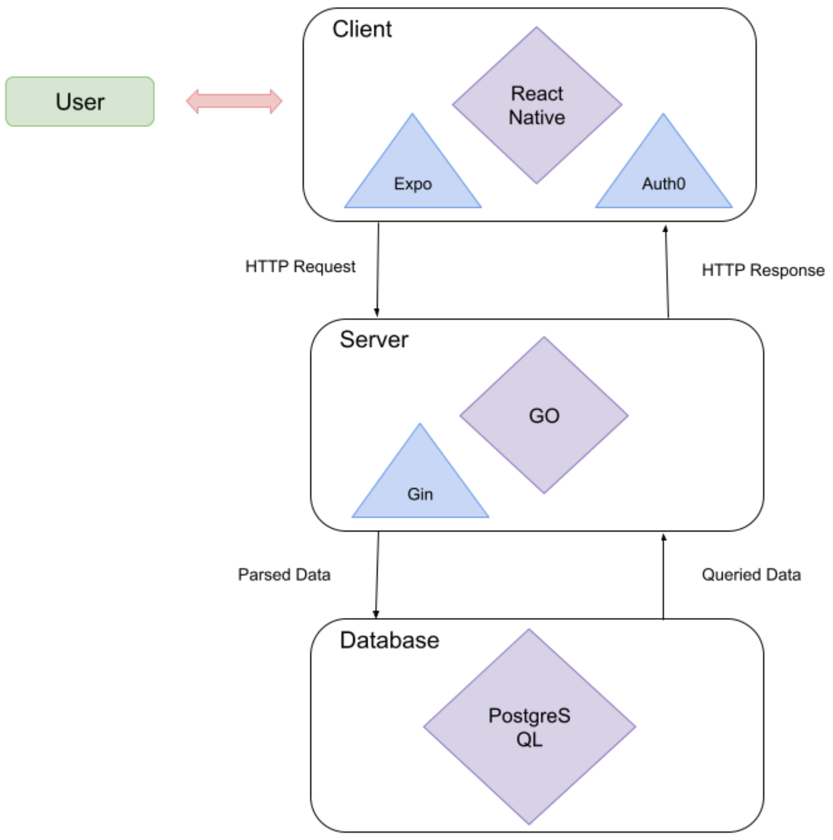
Handler Name: CourseHandler	
Responsibilities: - Handling all requests regarding the \`Course\` struct and returning the corresponding responses back to the client - GetCourse - CreateCourse - DeleteCourse - GetAllCourses - GetStudents - AddStudent - RemoveStudent	Collaborators: - User - Course - Jwt-go - Gin - net/http - encoding/json - Fmt - UserService - CourseService

Server

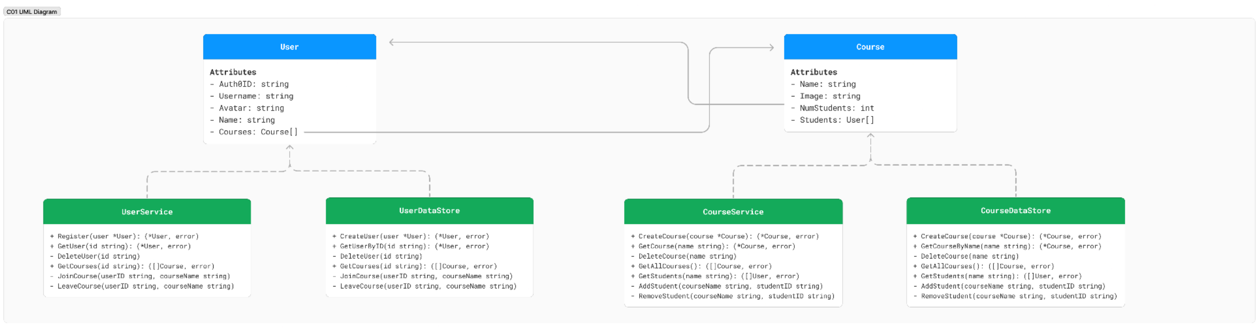
src/server/main.go

Name: Main	
Responsibilities: - Loading the db into the context - Loading the services in the router - Starting the server and starting the HTTP/TCP connection - Logging requests/responses	Collaborators: - Context - Log - net/http - Os - os/signal - Syscall - Time - Handler

System Architecture

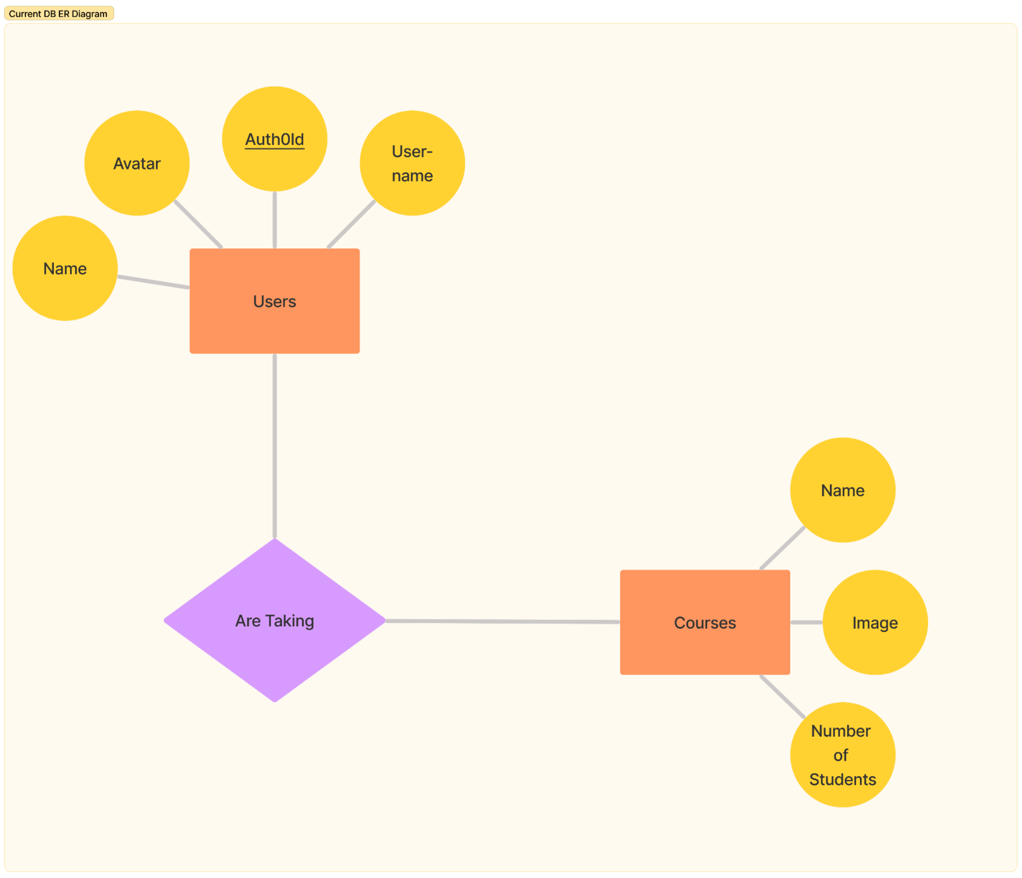


Backend System



```
User
Attributes
- Auth0ID: string
- Username: string
- Avatar: string
- Name: string
- Courses: Course[]

Course
Attributes
- Name: string
- Image: string
- NumStudents: int
- Students: User[]

UserService
+ Register(user *User): (*User, error)
+ GetUser(id string): (*User, error)
- DeleteUser(id string)
+ GetCourses(id string): ([]Course, error)
- JoinCourse(userID string, courseName string)
- LeaveCourse(userID string, courseName string)

UserDataStore
+ CreateUser(user *User): (*User, error)
+ GetUserByID(id string): (*User, error)
- DeleteUser(id string)
+ GetCourses(id string): ([]Course, error)
- JoinCourse(userID string, courseName string)
- LeaveCourse(userID string, courseName string)

CourseService
+ CreateCourse(course *Course): (*Course, error)
+ GetCourse(name string): (*Course, error)
- DeleteCourse(name string)
+ GetAllCourses(): ([]Course, error)
+ GetStudents(name string): ([]User, error)
- AddStudent(courseName string, studentID string)
- RemoveStudent(courseName string, studentID string)

CourseDataStore
+ CreateCourse(course *Course): (*Course, error)
+ GetCourseByName(name string): (*Course, error)
- DeleteCourse(name string)
+ GetAllCourses(): ([]Course, error)
+ GetStudents(name string): ([]User, error)
- AddStudent(courseName string, studentID string)
- RemoveStudent(courseName string, studentID string)
```

User-Course Relation Model



System Decomposition

The system contains three main parts: Client, Server, Database. The users engage with the `Client` through quick and seamless UI interactions. The full UI/UX is built in React Native to guarantee clean components and fast rendering. Through various Axios requests, the users are able to interact with the functionality provided by the `Server`. These include various CRUD operations, and in the future, a variety of communications, analytics, and content uploading/viewing. Our server is written in Go and utilises the `Gin` http framework to streamline the development of the API. Upon receiving data from the user, data is deserialized into structs which have guaranteed protection against various malicious inputs. In regards to errors, Go has the notion of treating errors as values, and all functions which can error, return an error as a value, which ensures that all errors are handled and detailed responses are given to the user if one were to occur.As for our database, the choice of PostgreSQL was made as our data is heavily relational so a tabular-store database was the obvious choice.