

StudyBuddies

SYSTEM DESIGN DOCUMENT

ASHWIN MAYURATHAN
BRANDAN BURGESS
CHINMAY GOKHALE
DAMIAN MELIA

Contents

Front-End

React Component(s): HomeScreen	3
React Component(s): AccountPage	3
React Component(s): NavBar	3

Back-End

Class/Structs:

DefaultModel	4
User	4
Course	4

Interfaces:

UserService	5
UserDataStore	5
CourseService	6
CourseDataStore	7

Handlers:

Handler	7
Handler	8
CourseHandler	9

Server:

Main.go	9
---------	---

Software Architecture

Three-Tier Architecture	10
System Decomposition	12

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

Backend

Classes/Structs

Class Name: DefaultModel	
Interfaces: None	
Responsibilities: <ul style="list-style-type: none">- Defines the basic `gorm.model` struct:<ul style="list-style-type: none">- ID: uint- Created At: timestamp- Updated At: timestamp-	Collaborators: <ul style="list-style-type: none">- None

Class Name: User	
Interfaces: UserService, UserDataStore	
Responsibilities: <ul style="list-style-type: none">- Simply defines the structure of the `User` data type<ul style="list-style-type: none">- DefaultModel- Auth0ID string- Username string- Avatar string- Name string- Courses []Course	Collaborators: <ul style="list-style-type: none">- DefaultModel

Class Name: Course	
Interfaces: UserService, UserDataStore	

Responsibilities: <ul style="list-style-type: none"> - Simply defines the structure and interfaces of the `Course` data type <ul style="list-style-type: none"> - DefaultModel - Name string - Image string - NumStudents int - Students []User 	Collaborators: <ul style="list-style-type: none"> - Defaultmodel
--	---

Interfaces

src/server/service/user_service.go

Interface Name: UserService	
Responsibilities: <ul style="list-style-type: none"> - Returns The Information gathered from the `data` layer Functions: <ul style="list-style-type: none"> - 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 	Collaborators: <ul style="list-style-type: none"> - User - Course

src/server/datastore/user_datastore.go

Interface Name: UserDataStore

Responsibilities:

- 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

Collaborators:

- 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, error)
- GetStudents(name string) ([]User, error)

Collaborators:

- User
- Course

<ul style="list-style-type: none"> - AddStudent(courseName string, studentID string) error - RemoveStudent(courseName string, studentID string) error 	
---	--

src/server/datastore/course_datastore.go

Interface Name: CourseDataStore	
Responsibilities: <ul style="list-style-type: none"> - All operations involving courses within the DB <ul style="list-style-type: none"> - 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: <ul style="list-style-type: none"> - User - Course - Errors - Gorm

Handlers

src/server/handlers/handler.go

Handler Name: Handler

Responsibilities: <ul style="list-style-type: none"> - Syncing the handlers with a `gin router` to be passed to the server - Creates the route groups: <ul style="list-style-type: none"> /api <ul style="list-style-type: none"> - /account <ul style="list-style-type: none"> - /register - /auth/callback - /login - /delete - /courses <ul style="list-style-type: none"> - /join - /leave - /course <ul style="list-style-type: none"> - /create - /delete - /students - /add_student - /del_student 	Collaborators: <ul style="list-style-type: none"> - User - Course - Jwt-go - Gin - net/http - encoding/json - Fmt - UserService - CourseService
--	---

src/server/handlers/user_handler.go

Handler Name: UserHandler	
Responsibilities: <ul style="list-style-type: none"> - Handling all requests regarding the `User` struct and returning the corresponding responses back to the client <ul style="list-style-type: none"> - AuthCallback (handles Auth0 token management) - Register - Login - Delete - GetCourses - JoinCourse - LeaveCourse 	Collaborators: <ul style="list-style-type: none"> - User - Course - Jwt-go - Gin - net/http - encoding/json - Fmt - UserService - CourseService

src/server/handlers/course_handler.go

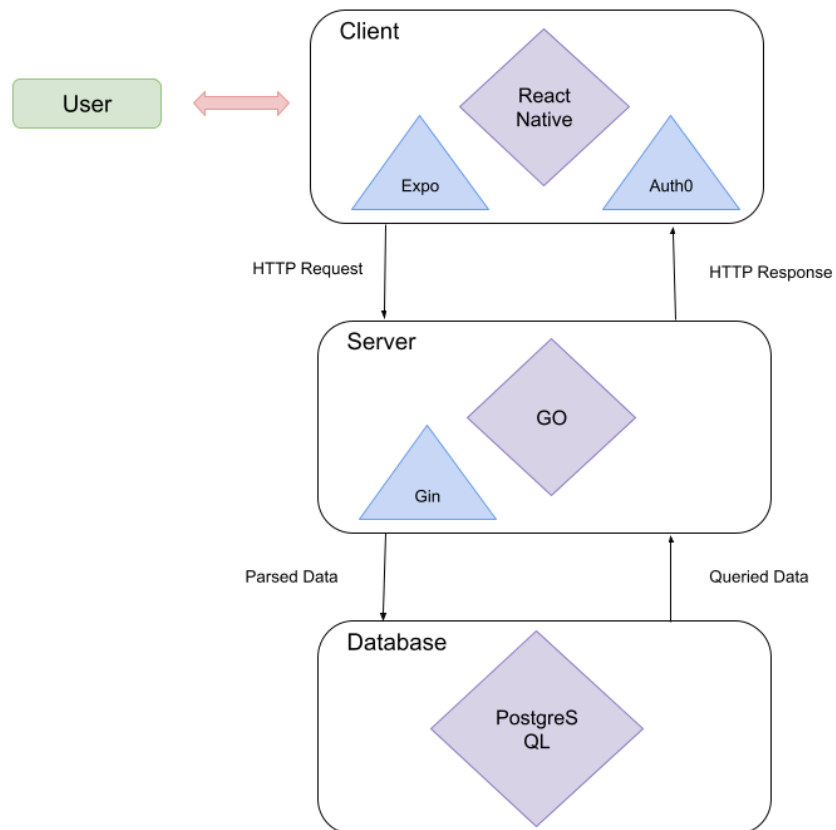
Handler Name: CourseHandler	
Responsibilities: <ul style="list-style-type: none">- Handling all requests regarding the `Course` struct and returning the corresponding responses back to the client- GetCourse- CreateCourse- DeleteCourse- GetAllCourses- GetStudents- AddStudent- RemoveStudent	Collaborators: <ul style="list-style-type: none">- User- Course- Jwt-go- Gin- net/http- encoding/json- Fmt- UserService- CourseService

Server

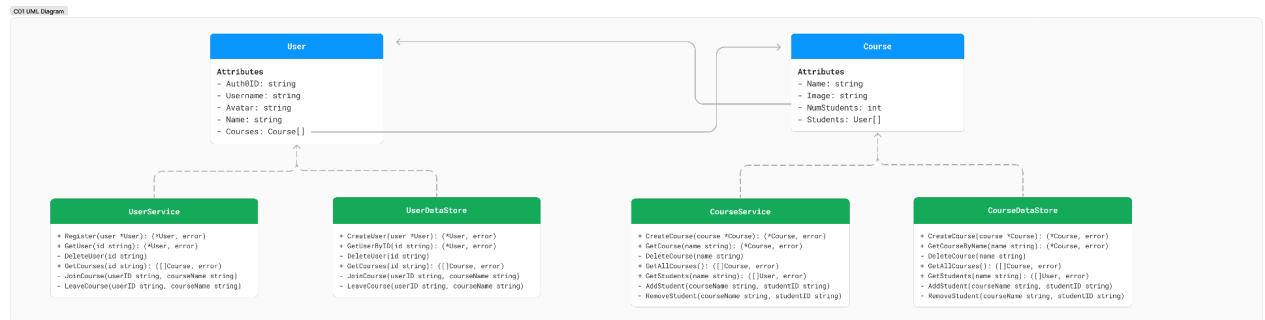
src/server/main.go

Name: Main	
Responsibilities: <ul style="list-style-type: none">- 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: <ul style="list-style-type: none">- Context- Log- net/http- Os- os/signal- Syscall- Time- Handler

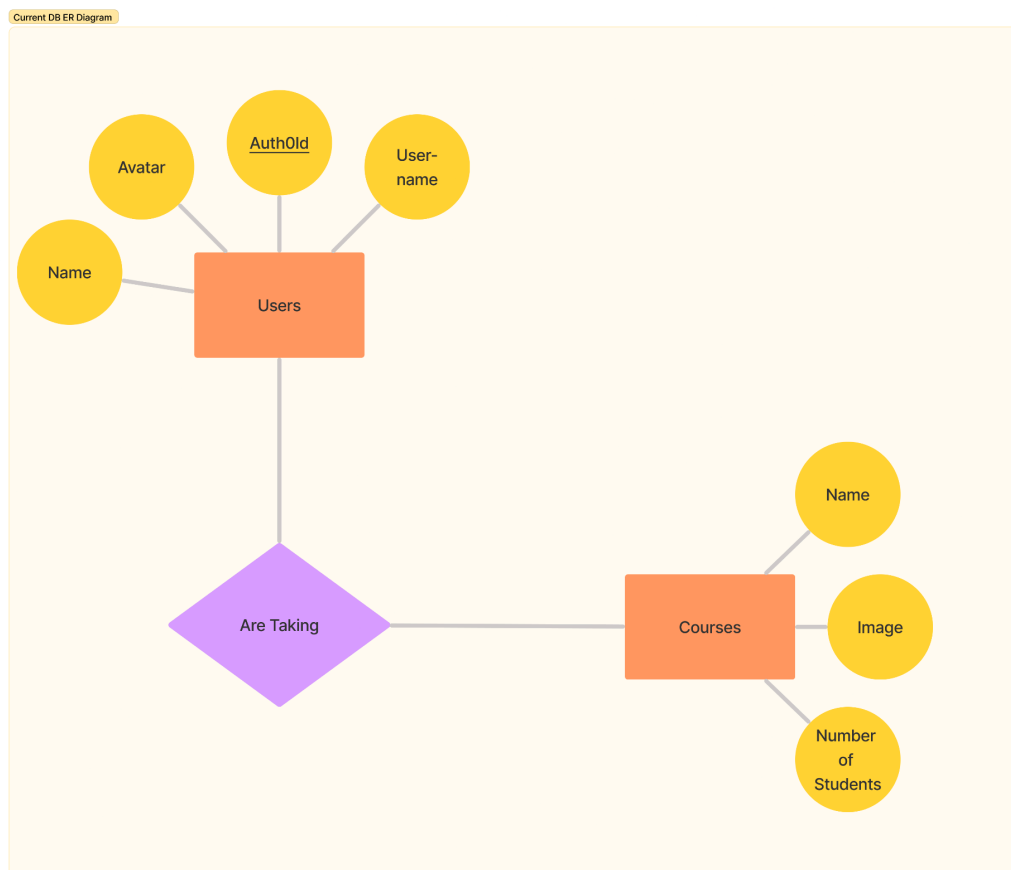
System Architecture



Backend System



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.