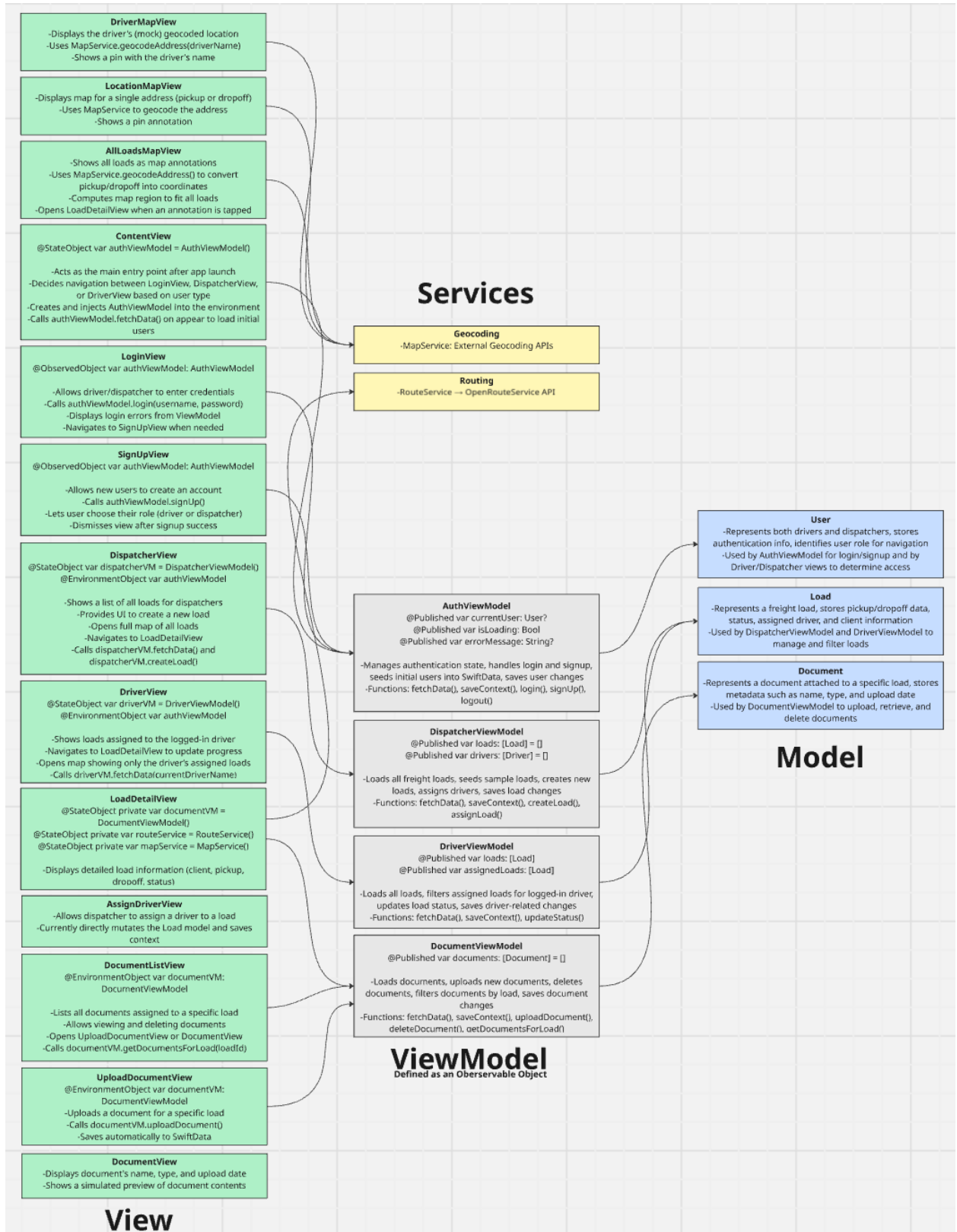# CSE 335 Class Project Phase III (Counted towards the Final Exam)
Submission Template

a)   Link for your technical presentation (Presentation length should not exceed 12 minutes and should not be less than 8 minutes).

b)   Copy and Paste your Final MVMM Architecture diagram. This diagram should be complete and functionality of each model, view, view-model(or controller) need to explained clearly showing how each component interact with each other in your application

View

DriverMapView
-Displays the driver's (mock) geocoded location
-Uses MapService.geocodeAddress(driverName)
-Shows a pin with the driver's name

LocationMapView
-Displays map for a single address (pickup or dropoff)
-Uses MapService to geocode the address
-Shows a pin annotation

AllLoadsMapView
-Shows all loads as map annotations
-Uses MapService.geocodeAddress() to convert pickup/dropoff into coordinates
-Computes map region to fit all loads
-Opens LoadDetailView when an annotation is tapped

ContentView
@StateObject var authViewModel = AuthViewModel()

-Acts as the main entry point after app launch
-Decides navigation between LoginView, DispatcherView, or DriverView based on user type
-Creates and injects AuthViewModel into the environment
-Calls authViewModel.fetchData() on appear to load initial users

LoginView
@ObservedObject var authViewModel: AuthViewModel

-Allows driver/dispatcher to enter credentials
-Calls authViewModel.login(username, password)
-Displays login errors from ViewModel
-Navigates to SignUpView when needed

SignUpView
@ObservedObject var authViewModel: AuthViewModel

-Allows new users to create an account
-Calls authViewModel.signUp()
-Lets user choose their role (driver or dispatcher)
-Dismisses view after signup success

DispatcherView
@StateObject var dispatcherVM = DispatcherViewModel()
@EnvironmentObject var authViewModel

-Shows a list of all loads for dispatchers
-Provides UI to create a new load
-Opens full map of all loads
-Navigates to LoadDetailView
-Calls dispatcherVM.fetchData() and dispatcherVM.createLoad()

DriverView
@StateObject var driverVM = DriverViewModel()
@EnvironmentObject var authViewModel

-Shows loads assigned to the logged-in driver
-Navigates to LoadDetailView to update progress
-Opens map showing only the driver's assigned loads
-Calls driverVM.fetchData(currentDriverName)

LoadDetailView
@StateObject private var documentVM = DocumentViewModel()
@StateObject private var routeService = RouteService()
@StateObject private var mapService = MapService()

-Displays detailed load information (client, pickup, dropoff, status)

AssignDriverView
-Allows dispatcher to assign a driver to a load
-Currently directly mutates the Load model and saves context

DocumentListView
@EnvironmentObject var documentVM: DocumentViewModel

-Lists all documents assigned to a specific load
-Allows viewing and deleting documents
-Opens UploadDocumentView or DocumentView
-Calls documentVM.getDocumentsForLoad(loadId)

UploadDocumentView
@EnvironmentObject var documentVM: DocumentViewModel
-Uploads a document for a specific load
-Calls documentVM.uploadDocument()
-Saves automatically to SwiftData

DocumentView
-Displays document's name, type, and upload date
-Shows a simulated preview of document contents

Services

Geocoding
-MapService: External Geocoding APIs

Routing
-RouteService → OpenRouteService API

ViewModel
Defined as an Oberservable Object

AuthViewModel: User?
@Published var currentUser: User?
@Published var isLoading: Bool
@Published var errorMessage: String?

-Manages authentication state, handles login and signup, seeds initial users into SwiftData, saves user changes
-Functions: fetchData(), saveContext(), login(), signUp(), logout()

DispatcherViewModel
@Published var loads: [Load] = []
@Published var drivers: [Driver] = []

-Loads all freight loads, seeds sample loads, creates new loads, assigns drivers, saves load changes
-Functions: fetchData(), saveContext(), createLoad(), assignLoad()

DriverViewModel
@Published var loads: [Load]
@Published var assignedLoads: [Load]

-Loads all loads, filters assigned loads for logged-in driver, updates load status, saves driver-related changes
-Functions: fetchData(), saveContext(), updateStatus()

DocumentViewModel
@Published var documents: [Document] = []

-Loads documents, uploads new documents, deletes documents, filters documents by load, saves document changes
-Functions: fetchData(), saveContext(), uploadDocument(), deleteDocument(), getDocumentsForLoad()

Model

User
-Represents both drivers and dispatchers, stores authentication info, identifies user role for navigation
-Used by AuthViewModel for login/signup and by Driver/Dispatcher views to determine access

Load
-Represents a freight load, stores pickup/dropoff data, status, assigned driver, and client information
-Used by DispatcherViewModel and DriverViewModel to manage and filter loads

Document
-Represents a document attached to a specific load, stores metadata such as name, type, and upload date
-Used by DocumentViewModel to upload, retrieve, and delete documents

c) Upload your PPT slides to the google drive and share the link for your PPT Slides

https://docs.google.com/presentation/d/1MHfd8rGJW_crEYg50bDJgLzJdQc9f1lC5vvMoLbdLtg/edit?usp=drive_link