EGE UNIVERSITY

FACULTY OF ENGINEERING

COMPUTER ENGINEERING DEPARTMENT

SOFTWARE ENGINEERING

2022–2023 FALL SEMESTER

CITY GUIDE APPLICATION ARCHITECTURAL MODEL

DELIVERY DATE

09/12/2022

PREPARED BY GROUP 32

05190000072, Tuğcan TOPALOĞLU

05180000070, Doğukan ARGÜÇ

05190000902, Lale Elif YEŞİL

05190000764, Osman CANBOĞA

İçindekiler

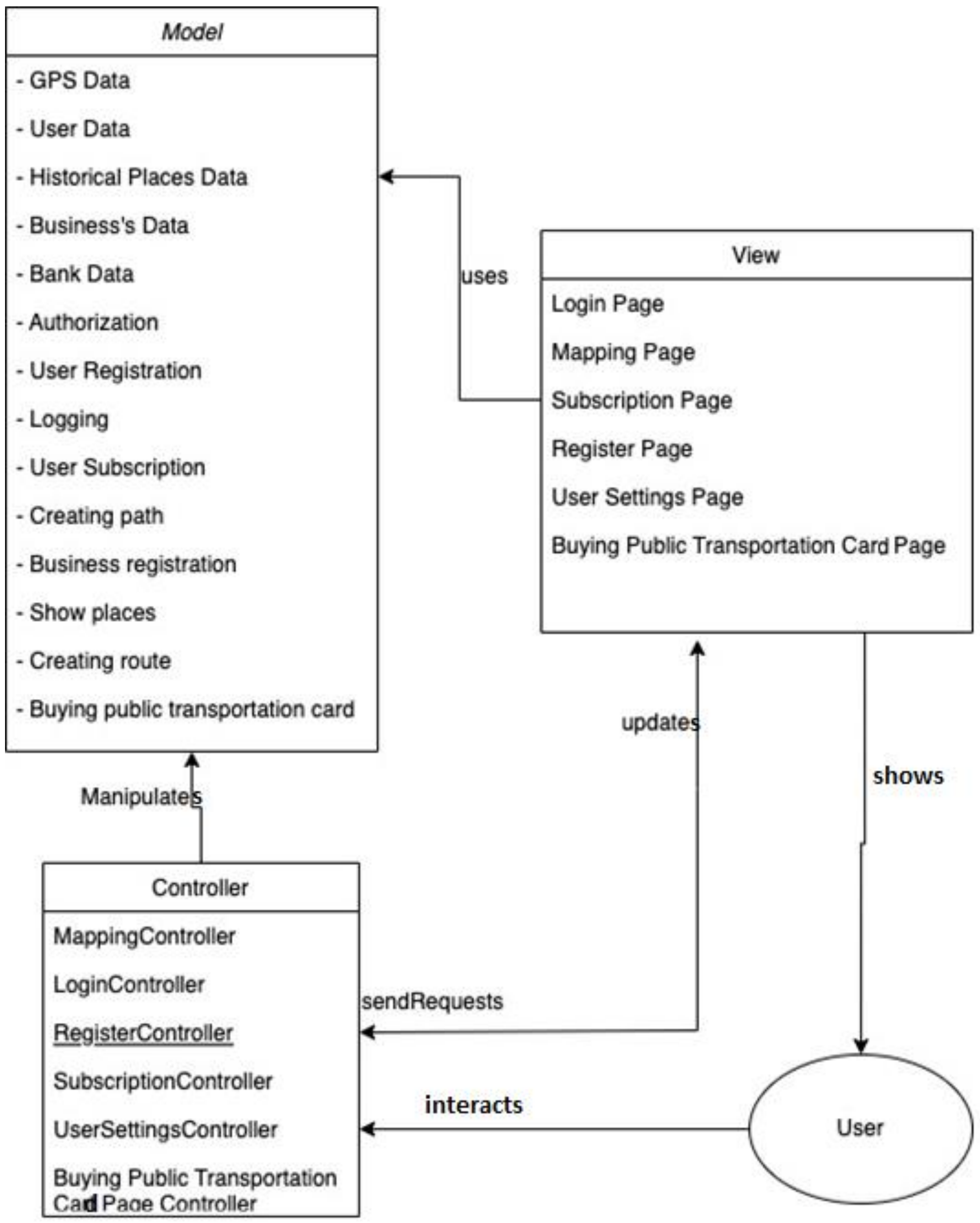
1) Introduction

We will choose and design the architectural model of our project during this section. We will choose the architectural style that best fits the needs of our program and then discuss why it is appropriate. We will describe the relationships, parts, and subsystems we employ. We'll evaluate it in comparison to other architectural designs. We'll describe the entities we employ.

2) Defining the Architecture

We chose mvc model. Its components are model, view and controller. Their sub-systems are for **model**: GPS Data, User Data, Historical Places Data, Business’s Data, Bank Data, Authorization, User Registration, Logging, User Subscription, Creating Path, Business Registration, Show Places, Creating Route, Buying Public Transportation Card; for **View**: Login Page, Mapping Page, Subscription Page, Register Page, User Settings Page, Buying Public Transportation Card Page; for **controller**: MappingController, LoginController, RegisterController, SubscriptionController, UserSettingsController, Buying Public Transportation Card Page Controller. The view uses model for database interaction. Shows itself to the User. Also, sends request to the controller. The controller subsystems manipulate model subsystems, and update view. The user interacts with controller.

3) Block Diagram



4) Explaining

Layering the subsystems and making them independent from each other is necessary for the correct and stable operation of our system. The "mvc" pattern is our best option, as we need to work more planned and concurrently than usual in our limited time. We didn’t choose the Layered architecture because due to the numerous levels of interpretation in our design, performance issues may arise. We couldn't choose the Repository architecture either as there is no need for storage all in one place. A simple error can affect the entire system, and we don’t want that during tourist travel or any time that she/he wants to eat, drink, buying public transport card, searching for accommodation.

5) Small Legend

Model

User Data: Data required to register the user to the system.

Historical Places Data: Data required to display historical places to the user

Business Data: Required data to show the features of the businesses to the user

Bank Data: Account data that must be obtained from the bank when the user becomes a member of the application and receives the bus card.

Authorization: Obtaining requested permissions such as KVKK, maps

User Registration: Receiving the necessary data for the user to register with the application

Logging: Retrieving the necessary data for the user to log into the application

User Subscription: Receiving the necessary data for the user to subscribe to the application

Creating Path: Creating a path for the user to go where they want

Business Registration: Businesses must register in order to exhibit in the application.

Creating Route: Creating a route for the subscribed user

Buying Public Transportation Card: The subscribed user receives a public transportation card through the application.

Show Places: Displaying historical places and businesses at user's choice

View

Login Page: The page where the user logs into the application

Mapping Page: Page showing places where the user can observe and visit herself on the map

Subscription Page: Page where user can purchase app subscription

Register Page: The page where the user registered in the application

User Settings Page: The page where the user can manage the settings related to the application

Buying Public Transportation Cart Page: The page where the user can pay for public transport and buy virtual cards through the application

Controller

MappingController: The controller that the mapping page controls.

LoginController: The controller that the Mapping page controls.

RegisterController: The controller that the Register page controls.

SubscriptionController: The controller that the Subscription page controls.

UserSettingsController: The controller that the UserSettings page controls.

BuyingPublicTransportationCartPageController: The controller that the BuyingPublictransportationCart page controls.

6) Conclusion

The architectural model that best fits our program was selected. We went into great length on why we settled on the MVC approach and why we rejected all other options. With the MVC approach, we created a block diagram and declared our entities.