

INTRODUCTION

Application Name: VivaVentura

Overview:

The goal is to design and develop an application called VivaVentura, which allows users to create, edit, and share detailed travel itineraries. VivaVentura will streamline the travel planning and management process, allowing users to manage trips, plan hotel stays, discover restaurants and attractions, and explore destinations. It's designed to make travel plans much easier to organize.

About:

VivaVentura is your companion for seamless travel planning and management. Whether you're embarking on a solo adventure, a romantic getaway, or a family vacation, VivaVentura empowers you to create, edit, and share detailed itineraries to make your trips unforgettable. This all-in-one travel app simplifies the entire travel experience from planning hotel stays to discovering the best places to eat, and exploring the cities and countries you'll visit.

VivaVentura is the perfect app for those seeking adventure. Share your adventures with the world and discover where others have been. Don't have a plan yet? Subscribe to VivaVentura's and discover the itinerary that best suits your stay.

Stakeholders:

Users and Travel Enthusiasts

Objectives:

- Enable users to create, manage, and share travel itineraries.
- Integrate with external services for hotel stays, restaurant reservations, and event bookings.
- Provide Google Maps integration for directions and navigation.
- Offer reminders for upcoming activities.
- Display ratings and information about stays, restaurants, and events pulled from Google.
- Implement subscription-based access for users.

Scope:

- Design and develop the VivaVentura app - high priority.
- Deliver a fully functional app for itinerary management - high priority.
- Implement Google Maps integration - High priority.
- Create a reminder system for upcoming activities - mid priority.
- Provide access to ratings and information from Google - mid priority.
- Integrate a subscription payment system - low priority.
- Integrate external services and conduct thorough testing - low priority.

Risks:

- Technical risks such as issues with external service integration or insufficient testing.
- Budget constraints may impact the app's feature set.
- Security concerns including data protection and secure payment processing.
- The app may not meet user expectations if not thoroughly tested and refined.

Planned Schedule:

Launch the app within 6 months.

Planned Budget:

Budget details to be determined.

High-Level View:

VivaVentura is designed for travelers who want to effortlessly plan and manage their trips. Users can create, edit, and share travel itineraries, incorporating hotel stays, dining experiences, and exploration of destinations. The app seamlessly integrates with external payment providers and security, offers Google Maps for navigation, sends reminders for scheduled activities, and provides information on activities and/or locations. A subscription payment system ensures users have access to features such as sharing and viewing other itineraries with more to come.

Additionally, VivaVentura will have a simple UI/UX to make travel planning enjoyable. Customer service providers will be 3rd parties, as the primary focus is on user-driven travel itinerary management. The success of the app will be measured by the delivery of a fully functional, user-friendly travel experience.

USE CASES

Epic:

As a customer, I should be able to create and plan a detailed itinerary of my trips.

Assumptions:

- The app will not handle reservations.
- Users will make in-app payments for subscriptions only that give them access to other features.
- Payments for subscriptions are handled by 3rd-party services.

User Story:

As a customer, I want to have a trip planning page where I can manage my trips.

Acceptance Criteria:

- Users can create detailed itineraries by adding destinations, dates, and activities.
- Itineraries can be named and customized..

User Story:

As a customer, I want to add timed and specific information about my trip into the itinerary.

Acceptance Criteria:

- Users can add schedules to their itineraries, specifying dates and times.
- The app integrates with external reservation systems for stays, restaurants, and events.

User Story:

As a customer, I want to navigate from my itinerary using Google Maps Integration.

Acceptance Criteria:

- Itineraries provide directions and maps using Google Maps.
- Users can view routes and navigate to their scheduled destinations.

User Story:

As a customer, I want to receive reminders of my upcoming trips and reservations from my itinerary.

Acceptance Criteria:

- Users receive reminders for upcoming activities in their itineraries.
- Reminders can be customized with notification preferences.

User Story:

As a customer, I want information about the destination or activities I have planned.

Acceptance Criteria:

- The app pulls ratings and information about stays, restaurants, and events from Google.
- Users can view details and reviews for each destination or activity.

User Story:

As a customer, I want the option to pay for other features that may be of use on my trip.

Acceptance Criteria:

- Users can subscribe to the app through an in-app payment system.
- Subscription options are clearly presented to users.

Epic:

As a user, I want a secure, customizable profile.

Assumptions:

- User information is stored securely.
- User profiles are protected with strong security measures.
- Two-factor security integration.

User Story:

As a user, I want to manage my profile and know my information is safe.

Acceptance Criteria:

- Users can create and edit their profiles.
- Profile information is securely stored and protected.

User Story:

As a user, I want my profile to be secure with strong password verification.

Acceptance Criteria:

- Users can securely sign-in and out of their accounts.
- Information is password-protected.
- User data is safeguarded from unauthorized access.
- Two-factor security is an option for users to turn on.

User Story:

As a user, I would like some form of customer service provided when I'm in a different timezone.

Acceptance Criteria:

- Users can contact a customer service number that will be open beyond the normal hours of the United States.
- AI chat-bot integration to assist navigating pages in the app.

Epic:

As a user, there should not be a lack of user experience optimization.

Assumptions:

- Assumes cloud optimization is used for workload performance.
- The app should provide a seamless user experience.
- It should handle user traffic effectively.

User Story:

As a user, I want a smooth performance when opening the app.

Acceptance Criteria:

- Users should not encounter significant delays or errors during app use.
- Interruptions should be brief and well-handled for a smooth user experience.

User Story:

As a user, data security should be prioritized to protect our information.

Acceptance Criteria:

- User information remains secure and protected.
- Robust security measures are in place to prevent unauthorized access.
- Any breaches should be handled immediately.
- Users should also be notified of breaches and requested to change passwords.

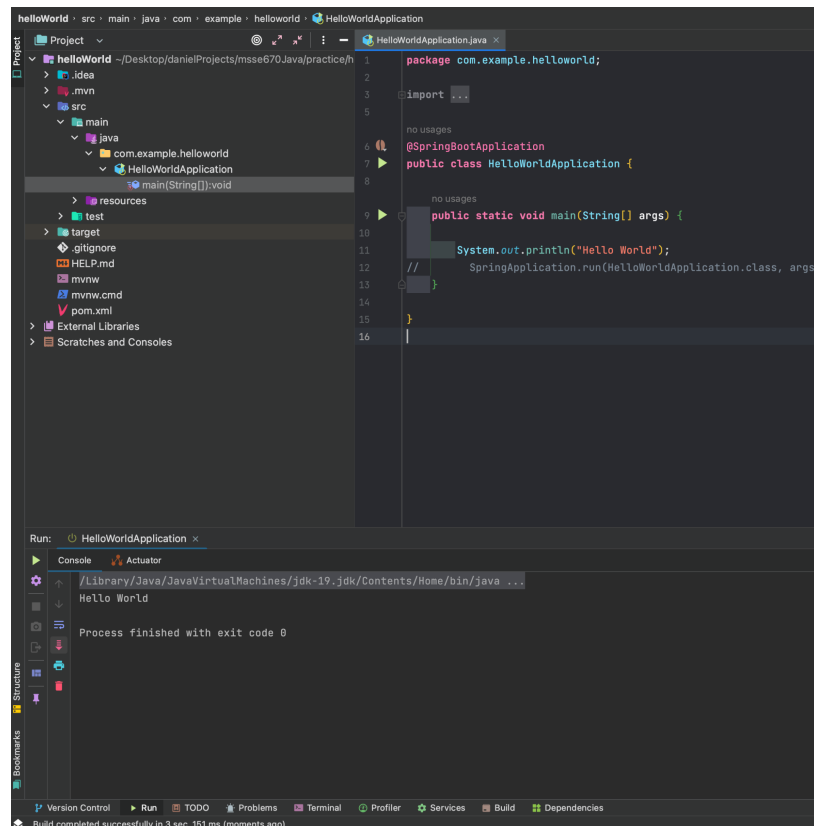
User Story:

As a user, the app should be highly available and scalable with downtime of less than 30 minutes.

Acceptance Criteria:

- The app should have high availability and remain responsive even during peak usage.
- Scalability should be implemented to handle increased user loads.

JAVA PROGRAM

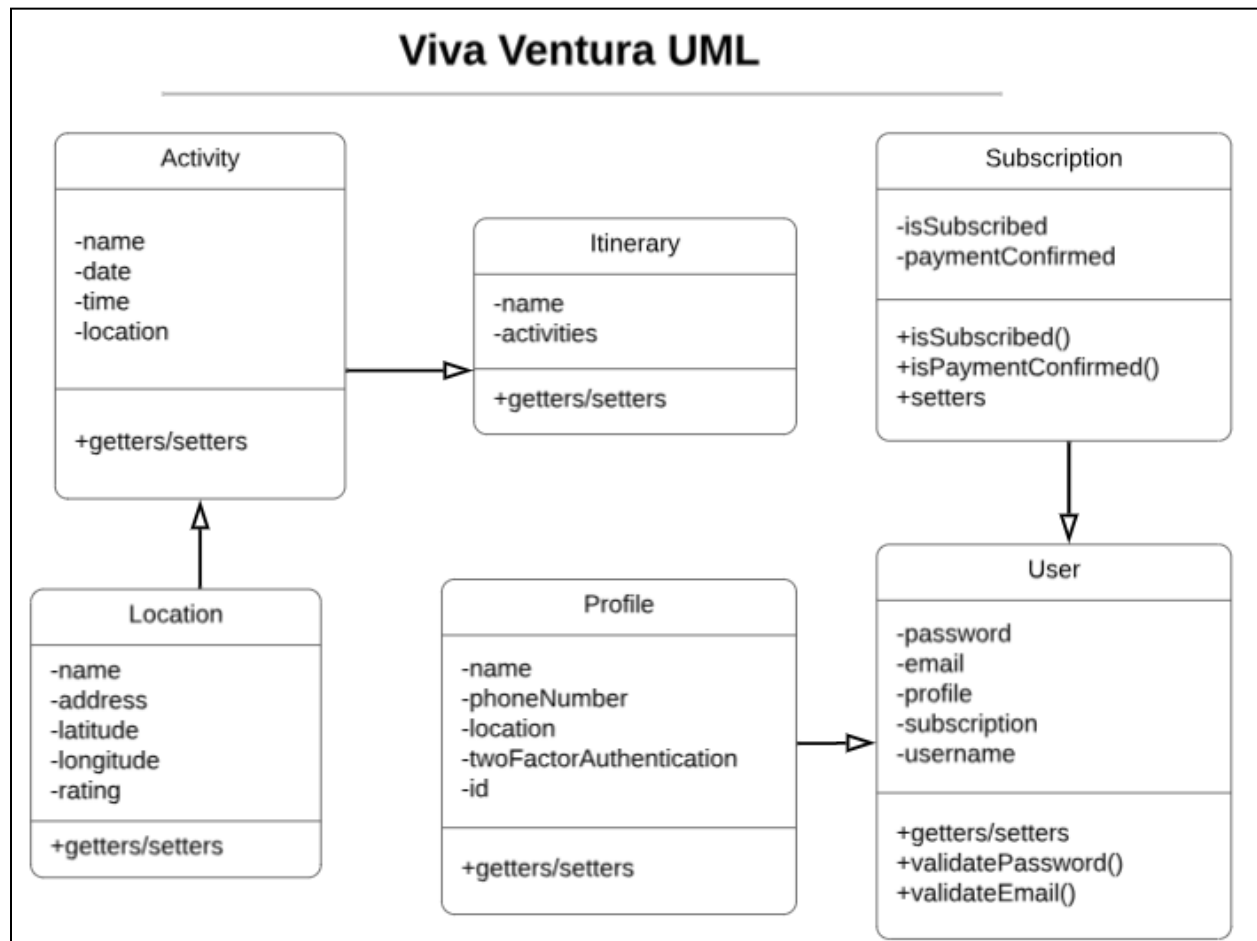


```
helloWorld · src · main · java · com · example · helloworld · HelloWorldApplication
Project
├── helloWorld
│   ├── .idea
│   ├── .mvn
│   ├── src
│   │   ├── main
│   │   │   ├── java
│   │   │   │   ├── com
│   │   │   │   │   ├── example
│   │   │   │   │   │   ├── helloworld
│   │   │   │   │   │   │   ├── HelloWorldApplication
│   │   │   │   │   │   │   │   ├── main(String[]):void
│   │   │   │   │   │   │   │   ├── resources
│   │   │   │   │   │   │   │   ├── test
│   │   │   │   │   │   │   │   ├── target
│   │   │   │   │   │   │   │   ├── gitignore
│   │   │   │   │   │   │   │   ├── HELP.md
│   │   │   │   │   │   │   │   ├── mvnw
│   │   │   │   │   │   │   │   ├── mvnw.cmd
│   │   │   │   │   │   │   │   ├── pom.xml
│   │   │   │   │   │   │   └── External Libraries
│   │   │   │   └── Scratches and Consoles
└── target
  ├── gitignore
  ├── HELP.md
  ├── mvnw
  ├── mvnw.cmd
  ├── pom.xml
  └── External Libraries
  └── Scratches and Consoles

no usages
6 @SpringBootApplication
7 public class HelloWorldApplication {
8
9     no usages
10    public static void main(String[] args) {
11
12        System.out.println("Hello World");
13        // SpringApplication.run(HelloWorldApplication.class, args);
14    }
15 }
16

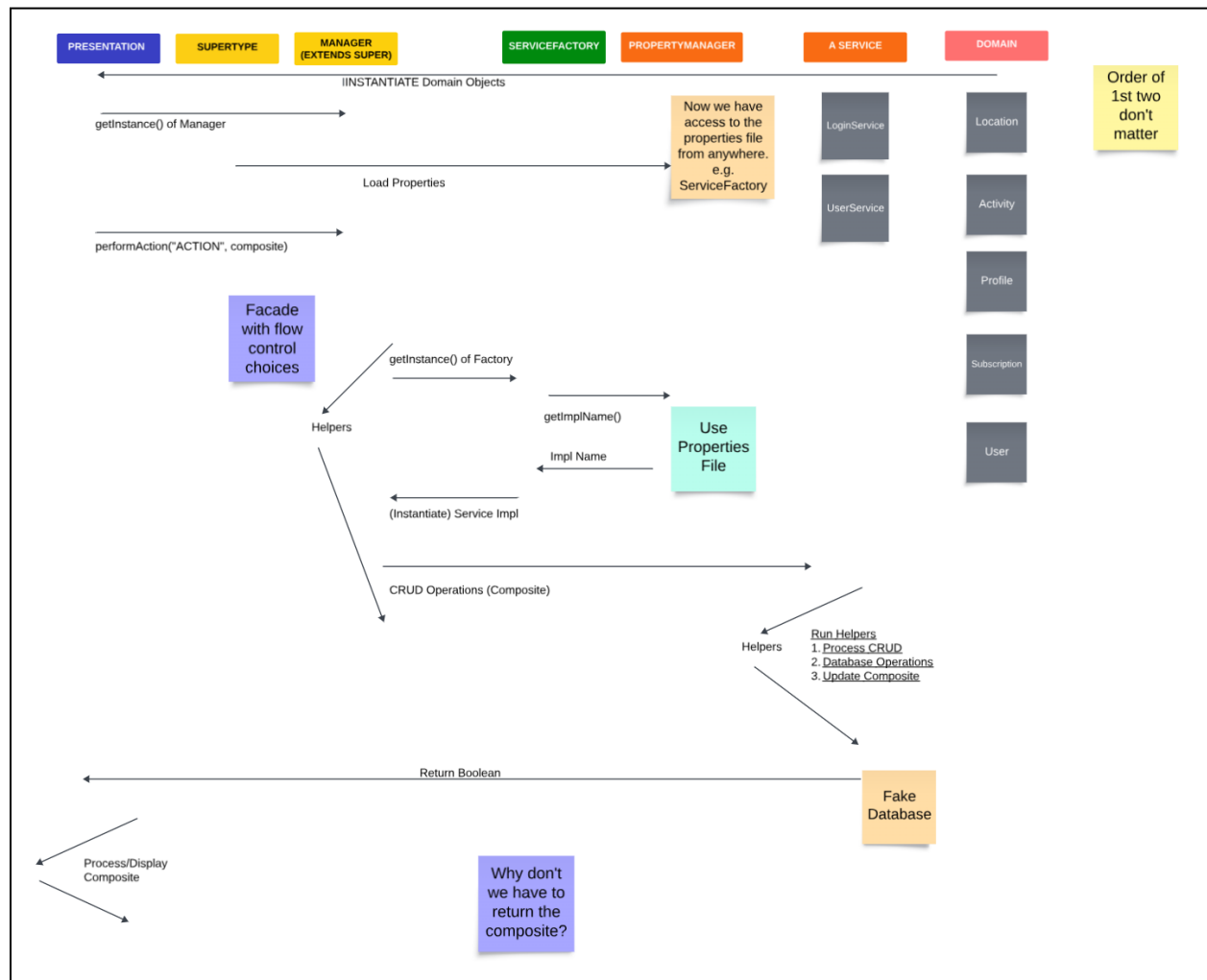
Run: HelloWorldApplication
Console
Actuator
/Library/Java/JavaVirtualMachines/jdk-19.jdk/Contents/Home/bin/java ...
Hello World
Process finished with exit code 0
Build completed successfully in 3 sec, 151 ms (moments ago)
```

UML MODEL



IMG: Here is a UML diagram of the Viva Ventura app. Most domain classes have only getters/setters and just a few will need custom methods like User class's validation methods.

SEQUENCE DIAGRAM



IMG: Sequence Diagram of the Viva Ventura app.

UNIT TESTS

```
class UserTest {  
    @Test  
    public void validateValidEmail() {  
        User user = new User();  
        assertTrue(user.validateEmail("test@example.com"));  
        assertTrue(user.validateEmail("user.name123@example.co.uk"));  
        assertTrue(user.validateEmail("john.doe123@gmail.com"));  
    }  
  
    @Test  
    public void validateInvalidEmail() {  
        User user = new User();  
        assertFalse(user.validateEmail("invalid-email.com"));  
        assertFalse(user.validateEmail("user@domain"));  
        assertFalse(user.validateEmail("missing@.com"));  
        assertFalse(user.validateEmail("user@example.com"));  
    }  
  
    @Test  
    public void validateValidPassword() {  
        User user = new User();  
        assertTrue(user.validatePassword("StrongP@ssw0rd"));  
        assertTrue(user.validatePassword("AnotherStrong123!"));  
    }  
  
    @Test  
    public void validateInvalidPassword() {  
        User user = new User();  
        assertFalse(user.validatePassword("WeakPass")); // Too short  
        assertFalse(user.validatePassword("nonunction123")); // Missing symbol  
    }  
}
```

```
//email validation method  
public boolean validateEmail(String email){  
    // Regular expression pattern for basic emails  
    String emailPattern = "[A-Za-z0-9+_-]*@[A-Za-z0-9+_-]*\\.([A-Za-z]{2,4})$";  
    Pattern pattern = Pattern.compile(emailPattern);  
    Matcher matcher = pattern.matcher(email);  
    // Checks if the email matches the emailPattern  
    return matcher.matches();  
}  
  
//password validation method  
public boolean validatePassword(String password){  
    // Check length of password  
    if (password.length() <= 12) {  
        System.out.println("Password is too short. It must be at least 12 characters long");  
        return false;  
    }  
  
    // Checking for characters in password  
    boolean hasUppercase = false;  
    boolean hasLowercase = false;  
    boolean hasDigit = false;  
    boolean hasSymbol = false;  
  
    // ForEach loop below takes the password & breaks each character into an array  
    // checking if it meets all the criteria */  
    for (char c : password.toCharArray()) {  
        //Below we use methods provided by the Character Class to check properties  
        if (Character.isUpperCase(c)) {  
            hasUppercase = true;  
        } else if (Character.isLowerCase(c)) {  
            hasLowercase = true;  
        } else if (Character.isDigit(c)) {  
            hasDigit = true;  
        } else if (Character.isSymbol(c)) {  
            hasSymbol = true;  
        }  
    }  
    return hasUppercase & hasLowercase & hasDigit & hasSymbol;  
}
```

Run: UserTest.validateInvalidEmail

Tests passed: 1 of 1 test - 18 ms

UserTest (com.vivaventura.model.domain, 18 ms)

validateInvalidEmail()

Process finished with exit code 0

IMG: Sample of 4 JUnit tests that validate valid and invalid email and passwords.