DESIGNING THE ARCHITECTURE

PRESENTED BY READML GROUP

Technology Stack

Layer	Technology
Backend	Java, Spring Boot
Database	PostgreSQL, JPA
Security	Spring Security, JWT
API	RESTful, Swagger
Build Tools	Maven
Deployment	Docker (если есть), Localhost

spring initializr

Project
- ○ Gradle - Groovy
- ○ Gradle - Kotlin   ● Maven
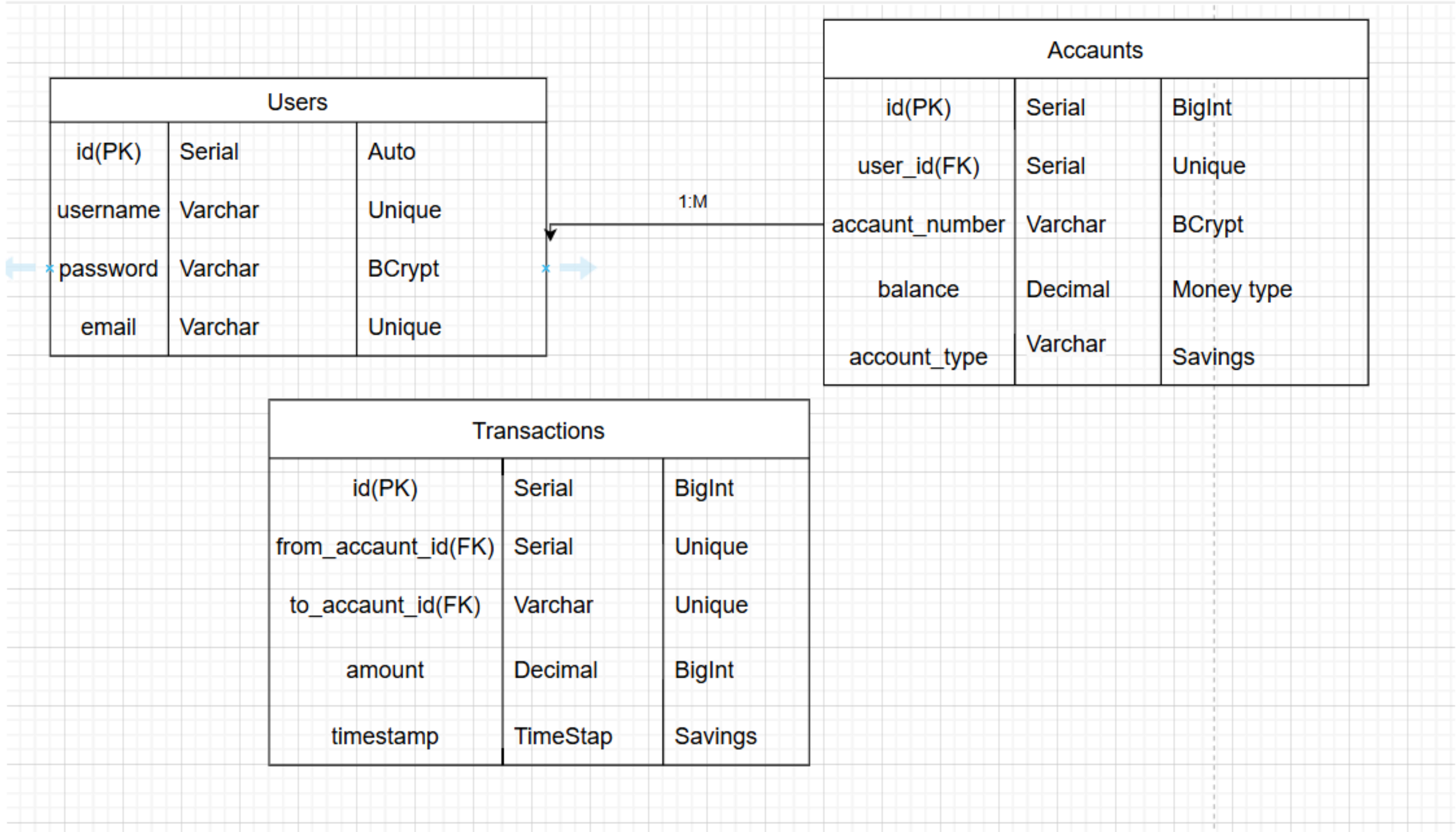
Language
- ● Java   ○ Kotlin   ○ Groovy

Spring Boot
- ○ 3.5.0 (SNAPSHOT)   ○ 3.5.0 (M2)   ○ 3.4.4 (SNAPSHOT)
- ● 3.4.3   ○ 3.3.10 (SNAPSHOT)   ○ 3.3.9

Our ER-DIAGRAMM

Users		
id(PK)	Serial	Auto
username	Varchar	Unique
password	Varchar	BCrypt
email	Varchar	Unique

1:M

Accaunts		
id(PK)	Serial	BigInt
user_id(FK)	Serial	Unique
accaunt_number	Varchar	BCrypt
balance	Decimal	Money type
account_type	Varchar	Savings

Transactions		
id(PK)	Serial	BigInt
from_account_id(FK)	Serial	Unique
to_accaunt_id(FK)	Varchar	Unique
amount	Decimal	BigInt
timestamp	TimeStap	Savings

users ↔ accounts:
one-to-many
relationship (one
user — many
accounts)

accounts ↔
transactions:
each transaction
refers to two
accounts (sender
and recipient

List of API Endpoints

3. Accounts (Account)

GET /accounts

Returns a list of all accounts.

GET /accounts/{id}

Returns details for a specific account.

POST /accounts

Creates a new account.

PUT /accounts/{id}

Updates an account (if necessary).

DELETE /accounts/{id}

Deletes an account (optional).

1. Authentication

GET /login

Returns the login page (Thymeleaf).

POST /login

Processes the login form (handled by Spring Security).

GET /register

Registration page (Thymeleaf).

POST /register

Creates a new user (via JSON or form submission).

2. Users (User)

GET /users

Returns a list of all users (JSON).

GET /users/{id}

Returns a user by ID.

POST /users

Creates a user (JSON).

PUT /users/{id}

Updates an existing user (JSON).

DELETE /users/{id}

Deletes a user by ID.

Brief Overview of the Architecture (MVC)

Controller

Receives HTTP requests (REST endpoints or Thymeleaf pages).
Calls the corresponding services.
Returns either JSON responses or rendered HTML templates.

Service

Contains the business logic (e.g., checking the account balance before a transaction).
Calls repositories for data storage and retrieval.
May invoke other services (e.g., an EmailService) if needed.

Repository (DAO)

Interfaces that extend Spring Data JPA (e.g., UserRepository, AccountRepository, TransactionRepository).
Responsible for CRUD operations on the database (create, read, update, delete).

Model (Entity)

Classes such as User, Account, and Transaction annotated with @Entity.
Fields correspond to columns in the database (e.g., id, username, balance, etc.).

Mockups

```java
13  public class AccountController {
14
15      @Autowired
16      private AccountService accountService;
17
18      @GetMapping
19      public List<Account> getAllAccounts() { return accountService.getAllAccounts(); }
22
23      @PostMapping
24      public ResponseEntity<Account> createAccount(@RequestBody Account account) {
25          Account newAccount = accountService.createAccount(account);
26          return ResponseEntity.ok(newAccount);
27      }
28  }
```

```java
public class DashboardController {

    @GetMapping("/dashboard")
    public String dashboard(@AuthenticationPrincipal org.springframework.secur:
                            Model model) {
        if (user != null) {
            model.addAttribute( attributeName: "username", user.getUsername());
        } else {
            model.addAttribute( attributeName: "username", attributeValue: "Guest");
        }
        return "dashboard";
    }
}
```

```java
public class RegistrationController {
    private final UserService userService;   2 usages

    public RegistrationController(UserService userService) { this.userS

    @GetMapping("/register")
    public String showRegistrationForm(Model model) {
        System.out.println("Открывается страница регистрации");
        model.addAttribute( attributeName: "user", new User());
        return "register";
    }


    @PostMapping("/register")
    public String processRegistration(@ModelAttribute("user") User user) {
        userService.createUser(user);
        return "redirect:/login";
    }
}
```

Mockups

```java
public class TransactionController {

    @Autowired
    private TransactionService transactionService;

    @GetMapping⊕⌄
    public List<Transaction> getAllTransactions() { return transactionService.getAllTransactions(); }


    @GetMapping(⊕⌄"/{id}")
    public Optional<Transaction> getTransactionById(@PathVariable Long id) {
        return transactionService.getTransactionById(id);
    }


    @PostMapping⊕⌄
    public Transaction createTransaction(@RequestBody Transaction transaction) {
        return transactionService.createTransaction(transaction);
    }
```

```java
public class Account {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne
    @JoinColumn(name = "user_id", nullable = false)
    private User user;

    @Column(nullable = false, unique = true)
    private String accountNumber;

    @Column(nullable = false)
    private Double balance;

    @Column(nullable = false)
    private String accountType; // "CHECKING" или "SAVINGS"

    @OneToMany(mappedBy = "fromAccount", cascade = CascadeType.ALL)
    private List<Transaction> sentTransactions;
```

```java
public class UserController {
    @Autowired
    private UserService userService;

    @GetMapping⊕⌄
    public List<User> getAllUsers() { return userService.getAllUsers(); }


    // Получить пользователя по ID
    @GetMapping(⊕⌄"/{id}")
    public ResponseEntity<User> getUserById(@PathVariable Long id) {
        Optional<User> user = userService.getUserById(id);
        return user.map(ResponseEntity::ok)
                .orElseGet(() -> ResponseEntity.notFound().build());
    }

    // Создать нового пользователя
    @PostMapping⊕⌄
    public ResponseEntity<User> createUser(@RequestBody User user) {
        User newUser = userService.createUser(user);
        return ResponseEntity.ok(newUser);
    }
}
```

RESULT
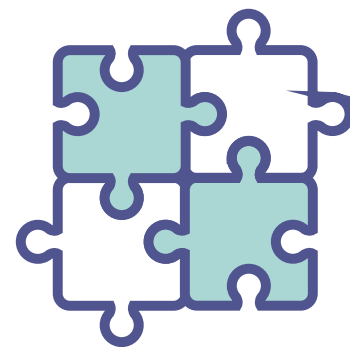
CREATE AN ACCOUNT

USERNAME

PASSWORD
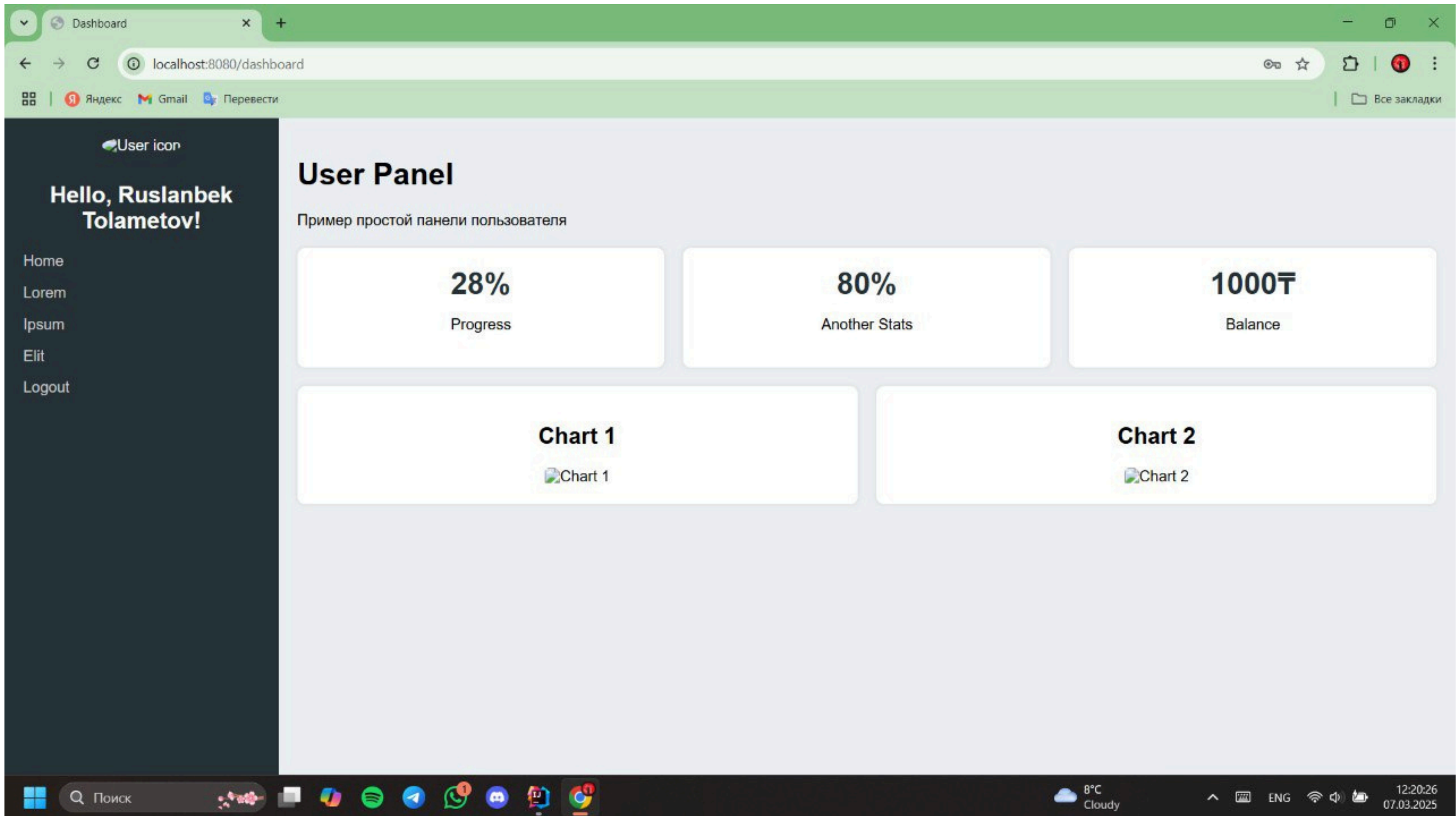
EMAIL

Register

Уже есть аккаунт? Войти

ACCOUNT LOGIN

USERNAME

PASSWORD

☐ Remember me   Forgot Password?

Login

Нет аккаунта? Зарегистрироваться

🌸User icon

Hello, Ruslanbek Tolametov!

Home

Lorem

Ipsum

Elit

Logout

User Panel

Пример простой панели пользователя

28%	**80%**	**1000₸**
Progress	Another Stats	Balance

Chart 1	**Chart 2**
🖼Chart 1	🖼Chart 2

Thank you very much!

PRESENTED BY READML GROUP