

# **ПРАКТИЧЕСКАЯ РАБОТА.**

## **ПОДХОДЫ ДЛЯ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

### **1.1. Цель и задачи практической работы**

Цель работы состоит в изучении и применении различных подходов к разработке программного обеспечения, основанного на тестировании, для повышения качества, надёжности и поддерживаемости кода.

Для достижения поставленной цели работы студентам необходимо выполнить ряд задач:

- 1) изучить теоретические основы методологий тестирования: TDD, ATDD, BDD и SDD;
- 2) исследовать преимущества и недостатки каждого подхода;
- 3) реализовать практические примеры для каждого метода;
- 4) проанализировать влияние интеграции тестирования на архитектуру и качество программного продукта;
- 5) подготовить итоговый отчёт с выводами и рекомендациями по интеграции подходов в современные процессы разработки.

### **1.2. Теоретический раздел**

#### ***1.2.1. Test-Driven Development (TDD)***

TDD, или Test-Driven Development (Разработка, управляемая тестами), — это методология разработки программного обеспечения, которая подразумевает создание тестов для функциональности ПО до того, как эта функциональность будет фактически реализована. TDD представляет собой циклический процесс, который помогает разработчикам создавать высококачественное, надежное ПО.

Ниже представлено подробное описание этапов разработки программного продукта при использовании подхода TDD.

1. На первой стадии создания программного продукта разработчик создает тесты, которые описывают ожидаемое поведение новой функциональности или компонента. Они обычно создаются на основе спецификаций, требований или ожиданий от заказчика.

2. На втором этапе после создания тестов разработчик запускает их. Поскольку код, реализующий требуемую функциональность, еще не написан, все тесты должны провалиться, что ожидаемо.

3. На третьем этапе разработчик начинает писать код, который будет реализовывать функциональность. Основная цель — написать минимальный объем кода, который позволит пройти тесты.

4. После написания некоторой части кода, разработчик снова запускает тесты, из которых один или несколько должны пройти, а некоторые могут оставаться проваленными.

После успешного прохождения тестов разработчик может провести рефакторинг кода. Его цель — улучшить читаемость, поддерживаемость и эффективность, но при этом не изменять функциональность будущего программного продукта.

5. Если приложение сложное, то процесс разработки будет циклическим, т.е. разработчик добавляет новые тесты, запускает их, реализует функциональность, запускает тесты снова, проводит рефакторинг и так далее. Цель — гарантировать, что каждая добавленная или измененная часть кода проходит тесты и не ломает то, что было реализовано ранее.

TDD является одним из ключевых инструментов для повышения качества программного обеспечения и ускорения процесса разработки.

Рассмотрим процесс разработки программного продукта методом TDD на примере создания простой функции для расчета факториала числа. Факториал числа  $n$  обозначается как  $n!$  и равен произведению всех целых чисел от 1 до  $n$ .

### **Шаг №1 Создание теста.**

Первый шаг в TDD — создание теста, описывающего ожидаемое поведение функции. В данном случае, создаётся тест для функции расчета факториала числа (рис. 3). Возможно использование различных библиотек для юнит-тестирования, например, `unittest` для Python.

```

import unittest

class TestFactorial(unittest.TestCase):
    def test_factorial_of_0(self):
        self.assertEqual(factorial(0), 1)

    def test_factorial_of_5(self):
        self.assertEqual(factorial(5), 120)

    def test_factorial_of_negative_number(self):
        with self.assertRaises(ValueError):
            factorial(-1)

if __name__ == '__main__':
    unittest.main()

```

*Рисунок 1. Пример теста для функции расчета факториала числа*

Было создано три теста:

- 1) 'test\_factorial\_of\_0' — проверяет, что факториал 0 равен 1.
- 2) 'test\_factorial\_of\_5' — проверяет, что факториал 5 равен 120.
- 3) 'test\_factorial\_of\_negative\_number' — проверяет, что функция вызывает исключение 'ValueError', если передано отрицательное число.

### **Шаг 2** Запуск тестов.

На этом этапе запускаются тесты. Так как функция 'factorial' еще не существует, все они провалятся.

### **Шаг 3** Реализация функции.

Создаётся функция 'factorial', чтобы сделать тесты успешными. Ниже на рис. 4 представлена её простая реализация.

```

def factorial(n):
    if n < 0:
        raise ValueError("Факториал не определен для отрицательных чисел")
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result

```

*Рисунок 2. Реализация функции факториала числа*

### **Шаг 4** Повторный запуск тестов.

Тесты запускаются повторно, чтобы убедиться, что функция factorial прошла успешно. Если что-то пошло не так, тесты сообщат о проблемах.

### **Шаг 5 Рефакторинг.**

После успешного прохождения тестов чаще всего необходимо провести рефакторинг кода, чтобы улучшить его читаемость и эффективность, при этом убедившись, что тесты остаются успешными. Наличие данного шага зависит от сложности разрабатываемого программного продукта.

Таким образом, TDD позволяет создавать функциональность шаг за шагом, начиная с определения ожидаемого поведения в виде тестов, что в свою очередь является залогом качественного кода с меньшим количеством ошибок.

### ***1.2.2. Acceptance Test-Driven Development (ATDD) — Разработка через тестирование приёмочных критериев***

ATDD — это методология, при которой разработка программного обеспечения начинается с формулирования приёмочных тестов, отражающих требования заказчика. Эти тесты пишутся совместно с участниками проекта (бизнес-аналитиками, заказчиками, тестировщиками и разработчиками) и определяют, какую функциональность должен предоставить продукт.

Ниже представлено подробное описание этапов разработки программного продукта при использовании подхода ATDD:

1. На начальном этапе все заинтересованные стороны собираются для обсуждения требований. Вместе они составляют список сценариев или тестовых случаев, описывающих, как должна работать система с точки зрения потребителя контента.
2. После того, как участники проекта (например, заказчик, бизнес-аналитик, тестировщики и разработчики) согласовали, как должна работать новая функциональность, требования оформляются в виде приёмочных сценариев. Далее они преобразуются в автоматизированные тесты, которые описывают, каким именно должно быть поведение системы при выполнении заданных условий. Поскольку в этот момент функциональность ещё не реализована, все тесты будут «падать».
3. Разработчики пишут код, обеспечивающий выполнение требований, зафиксированных в приёмочных тестах. Основная цель — реализовать функциональность так, чтобы при запуске тесты стали успешными.
4. После реализации кода тесты запускаются повторно. Если все тесты проходят, это свидетельствует о том, что функциональность соответствует ожиданиям, согласованным с заказчиком и бизнес-аналитиками.

5. При необходимости проводится рефакторинг, оптимизация кода и улучшение архитектуры, при этом тесты повторно запускаются, чтобы убедиться, что все приёмочные критерии сохранены.

Рассмотрим процесс разработки программного продукта методом ATDD на примере разработки функциональности регистрации пользователя.

### **Шаг 1**

На встрече с заказчиком и бизнес-аналитиками обсуждаются следующие требования. Например, если пользователь вводит уникальный Email и корректный пароль, регистрация проходит успешно и, если Email уже существует в системе, появляется сообщение об ошибке.

### **Шаг 2**

На основании этих требований создаются приёмочные тесты с использованием, например, фреймворка для API-тестирования, как показано ниже на рис. 5.

```
def test_successful_registration():
    response = register_user(email="user@example.com", password="SecurePass123")
    assert response.status_code == 200
    assert "confirmation" in response.body

def test_registration_with_existing_email():
    register_user(email="existing@example.com", password="Password1")
    response = register_user(email="existing@example.com", password="Password1")
    assert response.status_code == 400
    assert "email already exists" in response.body
```

*Рисунок 3. Пример приёмочных тестов для функции регистрации*

### **Шаг 3**

При запуске тестов — они проваливаются, так как функциональность ещё не реализована.

### **Шаг 4**

Разработчики пишут код регистрации (рис. 6), который проходит тесты.

```

# Простая имитация базы данных пользователей
users_db = {}

class Response:
    def __init__(self, status_code, body):
        self.status_code = status_code
        self.body = body

def register_user(email, password):
    """
    Регистрирует пользователя.
    Если email уже существует в базе, возвращает ошибку.
    Иначе сохраняет пользователя и имитирует отправку письма подтверждения.
    """
    # Проверка существования email в базе данных
    if email in users_db:
        return Response(400, "Error: email already exists")

    # Добавляем нового пользователя
    users_db[email] = {
        'password': password, # В реальном приложении пароль следует хранить в зашифрованном виде
        'confirmed': False    # Флаг подтверждения регистрации
    }

    # Имитируем отправку письма подтверждения (например, через email)
    # Здесь можно добавить логику отправки email
    confirmation_message = "Registration successful. Confirmation email sent."

    return Response(200, confirmation_message)

# Пример использования функции:
if __name__ == '__main__':
    # Первый вызов: регистрация нового пользователя
    response1 = register_user("user@example.com", "SecurePass123")
    print(f"Status: {response1.status_code}, Body: {response1.body}")

    # Попытка зарегистрировать пользователя с существующим email
    register_user("existing@example.com", "Password1") # первый вызов для создания записи
    response2 = register_user("existing@example.com", "Password1") # повторный вызов должен вернуть ошибку
    print(f"Status: {response2.status_code}, Body: {response2.body}")

```

*Рисунок 4. Пример реализации функции register\_user*

Код реализует функцию register\_user, которая проходит два приёмочных теста:

- 1) при успешной регистрации (новый Email) функция возвращает ответ с кодом 200 и сообщением о том, что регистрация прошла успешно (например, с отправкой письма подтверждения);
- 2) при попытке зарегистрировать пользователя с уже существующим Email функция возвращает ответ с кодом 400 и сообщением об ошибке.

## Шаг 5

Тесты повторно запускаются и успешно проходят, что гарантирует, что продукт соответствует согласованным требованиям.

Acceptance Test-Driven Development (ATDD) позволяет команде разработки, тестировщикам и бизнес-аналитикам совместно формировать требования к си-

стеме в виде приёмочных тестов ещё до написания кода. Это способствует лучшему пониманию ожидаемого поведения системы, повышает прозрачность разработки и снижает количество ошибок. В результате команда получает рабочий продукт, который точно соответствует бизнес-требованиям и успешно проходит автоматизированные тесты.

ATDD успешно применяется в различных реальных проектах. Например, в проекте WebWhiteboard.com была реализована функция защиты виртуальных досок паролем.

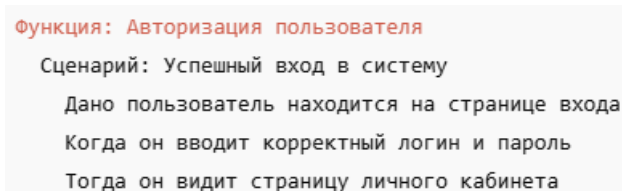
### 1.2.3. Behavior-Driven Development (BDD)

Behavior-Driven Development (BDD), или «разработка через поведение», — это методология разработки программного обеспечения, которая фокусируется на описании ожидаемого поведения системы с использованием понятного всем участникам процесса естественного языка. BDD является развитием подхода Test-Driven Development (TDD) и направлена на улучшение взаимодействия между разработчиками, тестировщиками и бизнес-аналитиками. В рамках BDD требования к системе формулируются в виде сценариев, описывающих конкретные примеры использования, что способствует созданию общего понимания и повышению качества разрабатываемого продукта.

Ниже представлено подробное описание этапов разработки программного продукта при использовании подхода BDD:

- 1) Формулирование сценариев на естественном языке. Они описываются с использованием структуры «Given–When–Then», где:
  - а) Given (Дано) — начальные условия или состояние системы;
  - б) When (Когда) — действие пользователя или событие;
  - в) Then (Тогда) — ожидаемый результат.

Пример описания сценария представлен на рис. 7.



Функция: Авторизация пользователя  
Сценарий: Успешный вход в систему  
Дано пользователь находится на странице входа  
Когда он вводит корректный логин и пароль  
Тогда он видит страницу личного кабинета

Рисунок 5. Пример описания сценария для авторизации пользователя

- 2) Сценарии превращаются в автоматизированные тесты с помощью инструментов (например, Cucumber, SpecFlow). На этом этапе сценарии запускаются — поскольку функциональность ещё не реализована, тесты проваливаются.

3) Разработчики пишут код, реализующий описанное поведение системы так, чтобы сценарии проходили успешно.

4) Сценарии запускаются снова, чтобы проверить, что система ведёт себя в соответствии с описанными ожиданиями.

5) При необходимости производится оптимизация кода, сопровождаемая повторным запуском тестов для подтверждения корректности изменений.

Рассмотрим процесс разработки программного продукта методом BDD для функции авторизации пользователя:

### **Шаг 1** Написание сценария.

В примере используется язык для формирования сценариев Gherkin (рис. 8).

```
Feature: User Registration

Scenario: Successful user registration
  Given the registration page is open
  When the user enters a valid email "user@example.com" and password "SecurePass123"
  And clicks the "Register" button
  Then the system should create a new account
  And show a success message "Registration successful. Confirmation email sent."

Scenario: Registration with an existing email
  Given the registration page is open
  And a user already exists with email "user@example.com"
  When the user tries to register with the same email "user@example.com"
  Then the system should display an error message "Error: email already exists"
```

*Рисунок 6. Описание сценария на языке Gherkin*

### **Шаг 2**

Автоматизация сценария с использованием, например, Java + Cucumber. В StepDefinitions.java прописаны шаги теста, эмулирующие регистрацию пользователя, проверку существующих email и вывод соответствующих сообщений (рис. 9). На данном этапе тест проваливается, так как система не реализует авторизацию.



```

import io.cucumber.java.en.*;
import org.junit.Assert;

public class StepDefinitions {
    private final UserService userService = new UserService();
    private String registrationMessage;

    @Given("the registration page is open")
    public void openRegistrationPage() {
        System.out.println("Registration page opened.");
    }

    @Given("a user already exists with email {string}")
    public void userAlreadyExists(String email) {
        userService.registerUser(email, "defaultPassword123");
    }

    @When("the user enters a valid email {string} and password {string}")
    public void enterUserCredentials(String email, String password) {
        registrationMessage = userService.registerUser(email, password);
    }
}

```

*Рисунок 7. Фрагмент кода теста*

### Шаг 3

Разработчики пишут функциональность, позволяющую проверить учетные данные и переход в личный кабинет (рис. 10).

```

import java.util.HashSet;
import java.util.Set;

public class UserService {
    private static final Set<String> registeredEmails = new HashSet<>();

    public String registerUser(String email, String password) {
        if (registeredEmails.contains(email)) {
            return "Error: email already exists";
        }
        registeredEmails.add(email);
        return "Registration successful. Confirmation email sent.";
    }

    public boolean isUserRegistered(String email) {
        return registeredEmails.contains(email);
    }
}

```

*Рисунок 8. Реализация логики регистрации*

#### **Шаг 4**

Тест запускается повторно и проходит успешно, подтверждая, что система ведёт себя согласно сценарию.

#### **Шаг 5**

При необходимости проводится рефакторинг без нарушения описанного поведения.

Методология BDD способствует улучшению коммуникации внутри команды и созданию общего понимания требований.

### ***1.2.4. Specification by Example (SDD)***

Specification by Example (SDD) — это подход, который использует конкретные примеры для описания требований и превращает их в автоматизированные тесты. Вместо абстрактных описаний бизнес-требований используются реальные примеры, позволяющие избежать неоднозначностей в понимании требований между бизнесом и разработчиками.

Ниже представлено подробное описание этапов разработки программного продукта при использовании подхода SDD.

Вместе с бизнес-аналитиками и заказчиками собираются конкретные примеры использования системы, которые отражают различные сценарии работы продукта. Примеры могут оформляться в виде таблиц, диаграмм или описаний.

Примеры превращаются в спецификации, которые описывают входные данные, ожидаемые результаты и условия работы. Эти спецификации служат основой для автоматизированных тестов.

Спецификации с примерами используются для написания автоматизированных тестов. На этом этапе тесты запускаются и проваливаются, поскольку функциональность ещё не реализована.

Разработчики пишут код, обеспечивающий выполнение спецификаций, чтобы все примеры, оформленные в виде тестов, проходили успешно.

После реализации функциональности тесты повторно запускаются для проверки корректности работы. Если требуется, проводится рефакторинг кода.

Рассмотрим процесс разработки программного продукта методом SDD для банковской системы, реализующей операцию снятия средств.

#### **Шаг 1 Документирование примеров.**

Документирование примеров представлено в виде табл. 1.

Таблица 1. Список примеров для банковской системы

Исходный баланс	Запрашиваемая сумма	Ожидаемый результат
1000	200	Баланс становится 800
500	600	Ошибка «недостаточно средств»

## Шаг 2

Создание спецификаций на основе таблицы, которые описывают поведение системы при попытке снять указанную сумму, представлено на рис. 11.

```
{
  "feature": "Транзакции банковского счета",
  "scenarios": [
    {
      "name": "Успешное снятие средств",
      "given": [
        "На счете имеется 1000 рублей"
      ],
      "when": "Пользователь запрашивает снятие 200 рублей",
      "then": [
        "Баланс становится 800 рублей"
      ]
    },
    {
      "name": "Недостаточно средств",
      "given": [
        "На счете имеется 500 рублей"
      ],
      "when": "Пользователь запрашивает снятие 600 рублей",
      "then": [
        "Система отображает ошибку 'недостаточно средств'"
      ]
    }
  ]
}
```

Рисунок 9. Создание спецификаций

## Шаг 3

Написание автоматизированных тестов (например, с использованием фреймворка для юнит-тестирования) представлено на рис. 12:

```
def test_withdrawal_success():
    account = BankAccount(balance=1000)
    new_balance = account.withdraw(200)
    assert new_balance == 800

def test_withdrawal_insufficient_funds():
    account = BankAccount(balance=500)
    try:
        account.withdraw(600)
    except InsufficientFundsError:
        pass
    else:
        assert False, "Ожидается исключение InsufficientFundsError"
```

*Рисунок 10. Примеры автоматизированных тестов*

## Шаг 4

Разработчики реализуют логику снятия средств, которая проходит вышеуказанные тесты (рис. 13).

```
class InsufficientFundsError(Exception):
    """Ошибка при недостатке средств на счете."""
    pass

class BankAccount:
    def __init__(self, balance: float):
        """Инициализация банковского счета с заданным балансом."""
        self.balance = balance

    def withdraw(self, amount: float) -> float:
        """Снятие средств со счета. Если средств недостаточно, выбрасывает исключение."""
        if amount > self.balance:
            raise InsufficientFundsError("Недостаточно средств")
        self.balance -= amount
        return self.balance
```

*Рисунок 11. Реализация логики снятия средств со счёта*

## Шаг 5

После успешного прохождения тестов при необходимости производится рефакторинг кода, гарантируя, что спецификации по-прежнему выполняются корректно.

Все четыре подхода — TDD, ATDD, BDD и SDD — интегрируют тестирование в процесс разработки, но делают это с разными акцентами:

- TDD базируется на тестах;

- ATDD фокусируется на согласовании приёмочных критериев с бизнесом и заказчиками до начала реализации;
- BDD использует сценарии на естественном языке для описания поведения системы;
- SDD строит спецификации на конкретных примерах, позволяющих устранить двусмысленность требований.

Каждая из методологий помогает обеспечить качество кода, уменьшить количество ошибок и улучшить коммуникацию между участниками проекта, выбирается в зависимости от специфики задачи и потребностей команды.

### 1.3. Описание работы

Практическая работа выполняется индивидуально.

Перед началом работы необходимо выбрать свой вариант задания. Он определяется по последним двум цифрам номера студенческого билета.

В процессе выполнения необходимо разработать программный продукт под Вашим номером, используя TDD, ATDD, BDD и SDD. В отчете необходимо подробно описать все этапы разработки, представленные выше в примерах.

#### **Варианты заданий**

##### **01. Калькулятор**

Функции — сложение, вычитание, умножение, деление.

TDD — написание тестов для арифметических операций, реализация минимального кода и рефакторинг.

ATDD — согласование приёмочных сценариев (например, «При вводе чисел и выборе операции результат должен быть корректным»).

BDD — сценарий «Given я ввожу 5 и 3, When я выбираю сложение, Then результат должен быть 8».

SDD — таблица с примерами входных данных и ожидаемых результатов.

##### **02. Конвертер единиц измерения**

Функции — преобразование температуры (Цельсий↔Фаренгейт), длины, веса.

TDD — тесты для корректности конвертации каждой единицы.

ATDD — приёмочные тесты для проверки пользовательских сценариев конвертации.

BDD — сценарий «Given я ввожу 100 Цельсий, When я выбираю конвертацию в Фаренгейт, Then я получаю 212°F».

SDD — спецификация с таблицами входных значений и результатов.

### **03. Приложение *To-Do List***

Функции — добавление задачи, удаление задачи, отметка выполненной задачи, просмотр списка.

TDD — тестирование каждой функции управления задачами.

ATDD — согласование сценариев работы пользователя (например, «При добавлении задачи она появляется в списке»).

BDD — сценарии на Gherkin для добавления, удаления и обновления статуса задач.

SDD — примеры использования с описанием входных данных и итогового состояния списка.

### **04. Приложение «Погодное оповещение»**

Функции — получение текущей погоды, прогноз на несколько дней, обновление информации, отправка оповещений.

TDD — тесты для функции получения и обновления данных о погоде.

ATDD — приёмочные тесты для сценариев оповещения (например, «Если температура ниже 0°C — отправить предупреждение»).

BDD — сценарии «Given актуальные данные, When наступает холод, Then пользователь получает уведомление».

SDD — таблицы с тестовыми данными для проверки алгоритма оповещения.

### **05. Банковский счёт**

Функции — пополнение, снятие средств, проверка баланса, перевод между счетами.

TDD — тестирование каждой банковской операции, обработка ошибок (например, недостаточно средств).

ATDD — согласование требований с «бизнесом» (например, «При попытке снятия суммы, превышающей баланс, должно возвращаться сообщение об ошибке»).

BDD — сценарий «Given баланс 2000, When снимается 200, Then баланс становится 1800».

SDD — спецификация операций с таблицами входных данных и ожидаемых результатов.

## **06. Система управления библиотекой**

Функции — добавление книги, поиск книги, выдача книги читателю, возврат книги.

TDD — тесты для каждой операции (например, проверка наличия книги, обработка ошибок при выдаче).

ATDD — приёмочные тесты, описывающие сценарии выдачи и возврата.

BDD — сценарий «Given книга доступна, When читатель запрашивает книгу, Then книга выдаётся».

SDD — примеры спецификаций с входными данными (ISBN, название) и результатами операций.

## **07. Викторина/Quiz приложение**

Функции — старт викторины, представление вопросов, приём ответов, расчёт баллов.

TDD — тестирование функций расчёта баллов и правильного переключения вопросов.

ATDD — сценарии «При правильном ответе баллы увеличиваются, при неправильном — нет изменения».

BDD — сценарии «Given вопрос задан, When пользователь отвечает правильно, Then баллы увеличиваются».

SDD — таблицы с вопросами, вариантами ответов и правильными результатами.

## **08. Чат-приложение (текстовый мессенджер)**

Функции — отправка сообщения, получение сообщения, отображение истории переписки.

TDD — тесты для отправки и получения сообщений, проверка истории.

ATDD — приёмочные тесты для сценариев общения.

BDD — сценарии «Given пользователь А отправил сообщение, When пользователь Б открывает чат, Then сообщение отображается».

SDD — примеры входных сообщений и ожидаемого отображения истории.

## **09. Конвертер валют**

Функции — получение курса валют, конвертация суммы, отображение истории конвертаций.

TDD — тесты для расчёта конвертации с различными курсами.

ATDD — приёмочные тесты, описывающие сценарии использования (например, «При вводе суммы и выборе валют результат должен соответствовать курсу»).

BDD — сценарий «Given курс валют актуален, When конвертирую 100 USD в EUR, Then получаю корректную сумму».

SDD — таблица курсов валют и тестовые примеры конвертации.

## **10. Управление инвентарём**

Функции — добавление товара, удаление товара, обновление количества, просмотр инвентаря.

TDD — тесты для операций с товарами, валидация входных данных.

ATDD — приёмочные тесты для сценариев обновления инвентаря.

BDD — сценарии «Given товар добавлен, When количество обновляется, Then инвентарь отражает изменения».

SDD — примеры спецификаций с таблицами товаров, количеств и операций.

## **11. Система голосования**

Функции — создание голосования, регистрация голосов, подсчёт голосов, отображение результатов.

TDD — тесты для корректного подсчёта голосов и обработки исключений.

ATDD — приёмочные тесты для сценариев голосования (например, «Каждый голос должен учитываться один раз»).

BDD — сценарий «Given голосование запущено, When участник голосует, Then голос учитывается».

SDD — таблицы с входными данными и ожидаемыми результатами голосования.

## **12. Приложение для учёта расходов**

Функции — добавление расхода, удаление записи, генерация отчёта по расходам.

TDD — тесты для корректного суммирования расходов и обработки ошибок.

ATDD — приёмочные тесты для сценариев добавления и отчёта расходов.

BDD — сценарии «Given запись добавлена, When генерирую отчёт, Then сумма расходов верна».

SDD — таблица с примерами расходов и итоговыми значениями.



### **13. Отдел кадров**

Функции — добавление сотрудника, поиск, обновление информации, удаление записи.

TDD — тесты для CRUD-операций с данными сотрудника.

ATDD — приёмочные тесты для сценариев управления сотрудниками.

BDD — сценарий «Given сотрудник добавлен, When ищу по фамилии, Then нахожу нужного сотрудника».

SDD — спецификации с примерами входных данных и ожидаемых результатов поиска.

### **14. Интернет-магазин (корзина покупок)**

Функции — добавление товара в корзину, удаление товара, расчёт итоговой суммы, оформление заказа.

TDD — тестирование расчёта итоговой суммы и операций с корзиной.

ATDD — приёмочные сценарии оформления заказа.

BDD — сценарий «Given товар в корзине, When оформляю заказ, Then итоговая сумма корректна».

SDD — таблицы с примерами товаров, ценами и итоговыми расчётами.

### **15. Система бронирования столиков в ресторане**

Функции — бронирование, отмена бронирования, просмотр текущих броней.

TDD — тесты для бронирования и отмены, проверка пересечений.

ATDD — приёмочные тесты для сценариев бронирования столика.

BDD — сценарий «Given свободный стол, When бронирую, Then бронь успешно оформлена».

SDD — примеры с таблицами времени, столиков и статусов бронирования.

### **16. Программа-будильник**

Функции — установка будильника, изменение времени, отключение сигнала.

TDD — тесты для установки и срабатывания будильника.

ATDD — приёмочные тесты для проверки работы будильника (например, «При наступлении времени – сигнал»).

BDD — сценарий «Given время установлено, When наступает время будильника, Then сигнал воспроизводится».

SDD — спецификации с примерами установки времени и ожидаемого поведения.

### **17. Менеджер рецептов**

Функции — добавление рецепта, поиск рецепта, отображение ингредиентов.

TDD — тесты для операций добавления и поиска рецептов.

ATDD — приёмочные сценарии для взаимодействия с рецептурой.

BDD — сценарий «Given рецепт сохранён, When ищу по названию, Then рецепт находится».

SDD — примеры рецептов с таблицами ингредиентов и шагов приготовления.

### **18. Музыкальный плеер**

Функции — воспроизведение, пауза, переключение треков.

TDD — тесты для проверки корректного переключения треков и управления воспроизведением.

ATDD — сценарии воспроизведения для проверки пользовательского опыта.

BDD — сценарий «Given треки загружены, When нажимаю «play», Then начинается воспроизведение».

SDD — примеры спецификаций с входными данными (список треков) и ожидаемыми состояниями плеера.

### **19. Видеоплеер/Библиотека видео**

Функции — просмотр списка видео, воспроизведение видео, поиск по заголовку.

TDD — тестирование функций поиска и воспроизведения.

ATDD — сценарии «При выборе видео оно начинается воспроизводиться».

BDD — сценарий «Given список видео, When выбираю видео, Then оно начинается воспроизводиться».

SDD — таблицы с примерами видео, названиями и статусами.

### **20. Блог-платформа**

Функции — создание поста, редактирование, удаление, комментирование.

TDD — тесты для CRUD-операций с постами и комментариями.

ATDD — приёмочные тесты для сценариев публикации и редактирования.

BDD — сценарий «Given пост создан, When редактирую, Then изменения сохраняются».

SDD — примеры спецификаций с форматами входных данных и результатами.

## **21. Планировщик задач**

Функции — добавление задачи, планирование, отметка выполнения.

TDD — тесты для добавления, изменения и удаления задач.

ATDD — сценарии для проверки корректности планирования задач.

BDD — сценарий «Given задача запланирована, When отмечаю выполнение, Then задача становится выполненной».

SDD — примеры спецификаций с таблицами задач и статусами.

## **22. Фитнес-трекер**

Функции — логирование тренировки, просмотр статистики, установка целей.

TDD — тесты для расчёта статистики и логирования данных.

ATDD — приёмочные тесты для сценариев отслеживания прогресса.

BDD — сценарий «Given тренировка записана, When запрашиваю статистику, Then вижу корректный результат».

SDD — спецификации с примерами тренировок и расчётами.

## **23. Менеджер паролей**

Функции — сохранение пароля, извлечение, обновление, удаление.

TDD — тесты для операций хранения и извлечения, обработка ошибок доступа.

ATDD — приёмочные тесты для сценариев защиты данных.

BDD — сценарий «Given пароль сохранён, When запрашиваю доступ, Then получаю пароль».

SDD — таблицы спецификаций с данными паролей и условиями доступа.

## **24. Шифратор файлов**

Функции — шифрование файла, дешифрование, удаление исходного файла.

TDD — тестирование корректности шифрования и дешифрования.

ATDD — сценарии для проверки безопасности (например, «После шифрования исходный файл недоступен»).

BDD — сценарий «Given файл существует, When его шифрую, Then могу расшифровать обратно в исходное состояние».

SDD — спецификации с примерами входных данных и результатами шифрования.

## **25. Сервис сокращения URL**

Функции — приём длинного URL, генерация короткого, перенаправление, статистика переходов.

TDD — тесты для генерации коротких URL и корректного перенаправления.

ATDD — приёмочные сценарии для проверки удобства использования сервиса.

BDD — сценарий «Given введён длинный URL, When нажимаю сокращение, Then получаю короткий URL, который работает».

SDD — таблицы с примерами URL и ожидаемыми результатами.

## **26. Социальная сеть (мини-версия)**

Функции — регистрация, добавление друзей, публикация статусов.

TDD — тесты для проверки регистрации, поиска пользователей и обновления статусов.

ATDD — приёмочные тесты для сценариев взаимодействия пользователей.

BDD — сценарий «Given зарегистрирован пользователь, When добавляю друга, Then друг отображается в списке».

SDD — спецификации с примерами входных данных (имена, email) и результатами.

## **27. Онлайн форум**

Функции — создание темы, публикация ответов, поиск по темам.

TDD — тесты для CRUD-операций с темами и ответами.

ATDD — сценарии для взаимодействия участников форума.

BDD — сценарий «Given тема создана, When публикую ответ, Then он отображается в теме».

SDD — примеры спецификаций с таблицами тем, ответов и поисковыми запросами.

## **28. Приложение для изучения флэш-карт**

Функции — создание флэш-карт, просмотр, отслеживание прогресса.

TDD — тесты для создания и обновления карточек.

ATDD — приёмочные сценарии для оценки эффективности обучения.

BDD — сценарий «Given флэш-карта создана, When начинаю сессию, Then карточка появляется для повторения».

SDD — таблицы с примерами вопросов, ответов и статистикой.

## **29. Календарь событий**

Функции — добавление события, редактирование, удаление, просмотр календаря.

TDD — тесты для операций с событиями и проверки дат.

ATDD — приёмочные тесты для сценариев планирования событий.

BDD — сценарий «Given событие запланировано, When просматриваю календарь, Then вижу событие в нужную дату».

SDD — спецификации с примерами дат и событий.

## **30. Приложение для заметок**

Функции — создание заметки, редактирование, удаление, поиск по заметкам.

TDD — тесты для проверки операций с заметками.

ATDD — приёмочные тесты для сценариев создания и поиска заметок.

BDD — сценарий «Given заметка создана, When ищу по ключевому слову, Then нахожу нужную заметку».

SDD — таблицы спецификаций с примерами заголовков, текста и результатов поиска.

## **31. Почтовый клиент**

Функции — отправка писем, получение писем, организация почтового ящика.

TDD — тесты для отправки/получения и сортировки писем.

ATDD — приёмочные тесты для сценариев использования почты.

BDD — сценарий «Given почтовый ящик настроен, When отправляю письмо, Then письмо доставляется».

SDD — спецификации с примерами писем и ожидаемым состоянием ящика.

## **32. Список покупок**

Функции — добавление товара, удаление, отметка купленного.

TDD — тесты для операций с элементами списка.

ATDD — сценарии для проверки корректности обновления списка.

BDD — сценарий «Given список пуст, When добавляю товар, Then он появляется в списке».

SDD — таблицы с примерами входных данных и итоговых списков.

### **33. Отслеживание привычек**

Функции — добавление привычки, отметка выполнения, просмотр статистики.

TDD — тесты для логики отметок и расчёта статистики.

ATDD — приёмочные тесты для проверки корректности отметок.

BDD — сценарий «Given привычка добавлена, When отмечаю её выполнение, Then статистика обновляется».

SDD — спецификации с примерами выполнения привычек.

### **34. Таймер медитации**

Функции — установка таймера, запуск, пауза, уведомление по окончании.

TDD — тесты для контроля работы таймера.

ATDD — сценарии для проверки корректности уведомлений.

BDD — сценарий «Given таймер установлен, When время истекает, Then появляется уведомление».

SDD — спецификации с примерами времени и ожидаемого поведения.

### **35. Переводчик текстов**

Функции — приём текста, перевод на выбранный язык, отображение результата.

TDD — тесты для проверки корректности перевода (с заглушками или API).

ATDD — приёмочные тесты для сценариев использования переводчика.

BDD — сценарий «Given текст введён, When выбираю язык, Then вижу переведённый текст».

SDD — таблицы с примерами исходного текста и ожидаемого перевода.

### **36. Карта/Навигатор**

Функции — поиск местоположения, построение маршрута, отображение маршрута.

TDD — тесты для проверки алгоритма построения маршрута.

ATDD — сценарии для подтверждения корректного отображения маршрута.

BDD — сценарий «Given введён адрес, When ищу маршрут, Then отображается оптимальный маршрут».

SDD — спецификации с примерами адресов и маршрутов.

### **37. Новостной агрегатор**

Функции — получение новостей, фильтрация по категориям, отображение списка новостей.

TDD — тесты для функций фильтрации и сортировки новостей.

ATDD — приёмочные тесты для сценариев выбора категории и просмотра новостей.

BDD — сценарий «Given новости загружены, When выбираю категорию «Спорт», Then вижу только спортивные новости».

SDD — таблицы с новостными данными и ожидаемыми результатами фильтрации.

### **38. Цифровые часы**

Функции — отображение текущего времени, смена формата времени, выбор часового пояса.

TDD — тесты для проверки корректного отображения времени.

ATDD — приёмочные тесты для сценариев смены формата.

BDD — сценарий «Given время отображается, When меняю часовой пояс, Then время корректно обновляется».

SDD — спецификации с примерами форматов времени и часовыми поясами.

### **39. Калькулятор индекса массы тела (BMI)**

Функции — приём роста и веса, вычисление BMI, категоризация результата.

TDD — тесты для вычисления и проверки градаций BMI.

ATDD — сценарии «Если BMI больше 25 — результат «Избыточный вес»».

BDD — сценарий «Given введены параметры, When рассчитывается BMI, Then выводится корректная категория».

SDD — таблицы с примерами роста, веса и ожидаемых категорий.

### **40. To-Do List с приоритетами**

Функции — добавление задачи с приоритетом, сортировка задач по приоритету, обновление статуса.

TDD — тесты для сортировки и обновления статусов задач.

ATDD — приёмочные сценарии для корректного отображения задач по приоритетам.

BDD — сценарий «Given задача с высоким приоритетом добавлена, When просматриваю список, Then задача отображается первой».

SDD — спецификации с примерами задач, приоритетов и итогового порядка.

#### **41. Напоминание о событиях**

Функции — добавление напоминания, уведомление пользователя, удаление напоминания.

TDD — тесты для функций установки и удаления напоминаний.

ATDD — приёмочные сценарии для проверки своевременных уведомлений.

BDD — сценарий «Given напоминание установлено, When наступает время, Then уведомление отображается».

SDD — примеры спецификаций с временами и ожидаемыми уведомлениями.

#### **42. Викторина с таймером**

Функции — запуск викторины, отслеживание времени на вопрос, подсчёт баллов.

TDD — тесты для контроля времени и подсчёта баллов.

ATDD — сценарии «При превышении времени ответ не засчитывается».

BDD — сценарий «Given вопрос задан, When время истекает, Then вопрос закрывается».

SDD — таблицы с примерами времени, ответов и баллов.

#### **43. Меню ресторана**

Функции — отображение меню, фильтрация по категориям, подробное описание блюд.

TDD — тесты для функций фильтрации и поиска блюд.

ATDD — сценарии для проверки корректного отображения информации о блюдах.

BDD — сценарий «Given меню загружено, When выбираю категорию «Напитки», Then вижу соответствующие позиции».

SDD — таблицы с описаниями блюд, категориями и ценами.

#### **44. Планировщик путешествий**

Функции — создание маршрута путешествия, добавление достопримечательностей, генерация маршрута.

TDD — тесты для функций создания маршрута.



ATDD — приёмочные сценарии для проверки корректного составления маршрута.

BDD — сценарий «Given выбран город, When добавляю достопримечательности, Then маршрут строится корректно».

SDD — спецификации с таблицами городов, достопримечательностей и маршрутов.

#### **45. *Онлайн-опрос/голосование***

Функции — создание опроса, сбор голосов, отображение результатов.

TDD — тесты для подсчёта голосов и обработки исключений.

ATDD — приёмочные сценарии для корректного голосования.

BDD — сценарий «Given опрос запущен, When голосую, Then голос учитывается».

SDD — таблицы с вариантами ответов и ожидаемыми результатами.

#### **46. *Приложение для прогноза погоды с оповещениями***

Функции — получение прогноза, установка оповещений, уведомление при изменениях.

TDD — тесты для функций оповещения и обновления прогноза.

ATDD — сценарии «При резком изменении температуры — отправить уведомление».

BDD — сценарий «Given прогноз обновился, When температура ниже порога, Then отправляется уведомление».

SDD — спецификации с таблицами изменений температуры и ожидаемыми оповещениями.

#### **47. *Органайзер музыкальной библиотеки***

Функции — добавление трека, редактирование метаданных, поиск по песням.

TDD — тесты для функций добавления и поиска треков.

ATDD — приёмочные сценарии для управления музыкальной коллекцией.

BDD — сценарий «Given трек добавлен, When ищу по названию, Then трек найден».

SDD — таблицы с метаданными треков и ожидаемыми результатами поиска.

#### **48. *Галерея изображений***

Функции — загрузка изображений, просмотр, удаление.

TDD — тесты для функций загрузки и удаления изображений.

ATDD — приёмочные тесты для сценариев работы с галереями.

BDD — сценарий «Given изображение загружено, When открываю галерею, Then изображение отображается».

SDD — спецификации с примерами изображений и операций.

#### **49. Приложение для цифровой подписи**

Функции — подписание документа, проверка подписи, хранение подписей.

TDD — тесты для проверки алгоритмов подписи и валидации.

ATDD — приёмочные сценарии для проверки безопасности подписи.

BDD — сценарий «Given документ готов, When подписываю, Then подпись валидна».

SDD — спецификации с примерами документов и ожидаемыми результатами проверки подписи.

#### **50. Онлайн-опросник**

Функции — создание опроса, сбор ответов, аналитика.

TDD — тесты для функций создания опроса и обработки ответов.

ATDD — сценарии для проверки корректности аналитики.

BDD — сценарий «Given опрос создан, When участник отвечает, Then данные обновляются».

SDD — таблицы с примерами вопросов, вариантов ответов и аналитическими данными.

#### **51. Виртуальная доска для рисования**

Функции — рисование, стирание, сохранение доски.

TDD — тесты для функций рисования и стирания.

ATDD — приёмочные сценарии для взаимодействия с интерфейсом.

BDD — сценарий «Given чистая доска, When рисую линию, Then линия появляется».

SDD — спецификации с примерами операций и итогового состояния доски.

#### **52. Приложение для разделения расходов**

Функции — добавление участников, распределение суммы, расчёт долей.

TDD — тесты для расчёта долей и распределения суммы.

ATDD — сценарии для проверки корректности распределения расходов.

BDD — сценарий «Given группа участников, When вводится сумма, Then рассчитываются доли».

SDD — таблицы с примерами расходов и ожидаемыми долями.

### **53. Рекомендательная система фильмов**

Функции — оценка фильмов, рекомендации, фильтрация по жанрам.

TDD — тесты для расчёта рекомендаций и обработки оценок.

ATDD — сценарии для проверки корректности рекомендаций.

BDD — сценарий «Given фильм оценён, When запрашиваю рекомендации, Then получаю похожие фильмы».

SDD — спецификации с таблицами оценок и результатов.

### **54. Система заказов в ресторане**

Функции — приём заказа, обновление статуса, генерация счёта.

TDD — тесты для операций заказа и обновления статуса.

ATDD — приёмочные тесты для сценариев оформления заказа.

BDD — сценарий «Given заказ создан, When обновляю статус, Then счёт генерируется».

SDD — таблицы с заказами, статусами и итоговыми расчётами.

### **55. Управление парковкой**

Функции — бронирование парковочного места, освобождение, просмотр доступности.

TDD — тесты для бронирования и освобождения мест.

ATDD — сценарии для проверки корректности управления парковкой.

BDD — сценарий «Given парковочное место свободно, When бронирую, Then место становится занятым».

SDD — спецификации с таблицами парковочных мест и статусами.

### **56. Система управления курсами (LMS)**

Функции — регистрация на курс, отслеживание прогресса, сдача тестов.

TDD — тесты для регистрации и отслеживания результатов тестов.

ATDD — приёмочные сценарии для проверки прохождения курса.

BDD — сценарий «Given студент зарегистрирован, When проходит тест, Then результат сохраняется».

SDD — спецификации с примерами курса, тестовых заданий и результатов.

### **57. Онлайн-аукцион**

Функции — листинг товаров, приём ставок, определение победителя.

TDD — тесты для корректного приёма ставок и определения победителя.

ATDD — сценарии для проверки сценария торгов.

BDD — сценарий «Given аукцион запущен, When участник делает ставку, Then ставка учитывается».

SDD — таблицы со ставками и итоговыми результатами.

### **58. Личный дневник**

Функции — создание записи, редактирование, поиск по записям.

TDD — тесты для операций с записями дневника.

ATDD — приёмочные сценарии для сценариев поиска и редактирования.

BDD — сценарий «Given запись создана, When ищу по ключевому слову, Then нахожу запись».

SDD — спецификации с примерами записей и критериями поиска.

### **59. Цифровое голосование**

Функции — создание выборов, регистрация голосов, подсчёт результатов.

TDD — тесты для подсчёта голосов и обработки ошибок.

ATDD — сценарии для проверки корректного голосования.

BDD — сценарий «Given выборы открыты, When голосую, Then голос учитывается».

SDD — таблицы с вариантами голосования и ожидаемыми итогами.

### **60. Тайм-трекер задач**

Функции — запуск таймера для задачи, остановка, отчёт по времени.

TDD — тесты для контроля времени работы над задачей.

ATDD — приёмочные тесты для проверки корректного подсчёта времени.

BDD — сценарий «Given задача начата, When останавливаю таймер, Then время фиксируется».

SDD — спецификации с примерами длительности задач и итоговых отчётов.

### **61. Планировщик бюджета**

Функции — установка бюджета, учёт расходов, генерация отчётов.

TDD — тесты для операций учёта и расчёта оставшихся средств.

ATDD — сценарии для проверки корректного учёта бюджета.

BDD — сценарий «Given бюджет установлен, When регистрирую расход, Then оставшаяся сумма пересчитывается».

SDD — таблицы с примерами доходов, расходов и итоговых сумм.

## **62. Обмен валют**

Функции — конвертация суммы, обновление курса, история конвертаций.

TDD — тесты для расчёта конвертации.

ATDD — приёмочные сценарии для проверки корректности обмена.

BDD — сценарий «Given курс обновлён, When конвертирую сумму, Then получаю корректное значение».

SDD — таблицы с курсами и примерами конвертации.

## **63. Медицинская запись (запись на приём)**

Функции — бронирование приёма, отмена, уведомление пациента.

TDD — тесты для функций бронирования и отмены.

ATDD — сценарии для проверки корректности уведомлений.

BDD — сценарий «Given свободное время, When пациент записывается, Then встреча подтверждается».

SDD — спецификации с примерами расписания и статусами встреч.

## **64. Система обратной связи**

Функции — отправка отзыва, просмотр отзывов, фильтрация по категориям.

TDD — тесты для функций отправки и фильтрации отзывов.

ATDD — приёмочные сценарии для проверки работы системы обратной связи.

BDD — сценарий «Given отзыв отправлен, When ищу по ключевому слову, Then отзыв находится».

SDD — таблицы с примерами отзывов и критериями фильтрации.

## **65. Трекер спортивных результатов**

Функции — добавление результатов, обновление таблицы, вывод рейтингов.

TDD — тесты для корректного обновления таблицы результатов.

ATDD — сценарии для проверки отображения актуальных результатов.

BDD — сценарий «Given результат игры внесён, When запрашиваю рейтинг, Then результаты отображаются корректно».

SDD — таблицы с результатами и рейтингами.

## **66. Онлайн-бронирование билетов**

Функции — выбор билета, бронирование, отмена бронирования, просмотр расписания.

TDD — тесты для функций бронирования и отмены.

ATDD — сценарии для проверки корректного отображения расписания.

BDD — сценарий «Given билет выбран, When бронирую, Then билет закрепляется за пользователем».

SDD — спецификации с примерами расписаний и статусов билетов.

## **67. Викторина для e-learning**

Функции — генерация викторины, приём ответов, оценка результатов.

TDD — тесты для функций генерации вопросов и оценки.

ATDD — сценарии для проверки корректности подсчёта баллов.

BDD — сценарий «Given викторина начата, When отвечаю на вопросы, Then результат рассчитывается».

SDD — таблицы с вопросами, вариантами ответов и правильными ответами.

## **68. Платформа для пожертвований**

Функции — приём пожертвований, отображение истории, генерация квитанций.

TDD — тесты для функций приёма средств и генерации квитанций.

ATDD — приёмочные сценарии для проверки корректности транзакций.

BDD — сценарий «Given пожертвование сделано, When запрашиваю квитанцию, Then она генерируется».

SDD — таблицы с транзакциями и примерами квитанций.

## **69. Система объявлений о недвижимости**

Функции — добавление объявления, поиск, фильтрация по параметрам.

TDD — тесты для операций добавления и поиска.

ATDD — сценарии для проверки корректной фильтрации объявлений.

BDD — сценарий «Given объявление добавлено, When ищу по параметру «количество комнат», Then нахожу соответствующие объекты».

SDD — спецификации с таблицами объявлений и фильтрами.

## **70. Каталог продуктов**

Функции — добавление продукта, поиск, фильтрация по категории.

TDD — тесты для операций с продуктами.

ATDD — сценарии для проверки работы каталога.

BDD — сценарий «Given продукт добавлен, When ищу по названию, Then он находится».

SDD — спецификации с примерами продуктов, категорий и результатов поиска.

### **71. Модерация форума**

Функции — флагирование сообщений, одобрение, удаление.

TDD — тесты для проверки функций модерации.

ATDD — сценарии для проверки работы модератора.

BDD — сценарий «Given сообщение помечено на модерацию, When одобряю, Then сообщение появляется на форуме».

SDD — таблицы с примерами сообщений и статусами модерации.

### **72. Отслеживание багов**

Функции — создание отчёта о баге, изменение статуса, закрытие.

TDD — тесты для операций с баг—трекером.

ATDD — сценарии для проверки жизненного цикла бага.

BDD — сценарий «Given баг зарегистрирован, When обновляю статус, Then статус меняется корректно».

SDD — таблицы с примерами багов и ожидаемыми статусами.

### **73. Анализ опросов**

Функции — создание опроса, сбор ответов, анализ данных.

TDD — тесты для функций обработки и анализа данных.

ATDD — Приёмочные сценарии для проверки корректности аналитики.

BDD — сценарий «Given опрос завершён, When анализирую результаты, Then данные сводятся корректно».

SDD — спецификации с примерами вопросов, ответов и аналитическими таблицами.

### **74. Медицинские записи**

Функции — добавление записи, обновление, поиск по пациентам.

TDD — тесты для операций с медицинскими данными.

ATDD — сценарии для проверки корректности поиска записей.

BDD — сценарий «Given запись пациента добавлена, When ищу по фамилии, Then запись находится».

SDD — спецификации с примерами записей и критериями поиска.

### **75. Расписание фитнес-классов**

Функции — создание расписания, регистрация участников, отмена занятия.

TDD — тесты для функций создания и обновления расписания.

ATDD — сценарии для проверки корректности регистрации.

BDD — сценарий «Given класс запланирован, When участник регистрируется, Then он отображается в списке».

SDD — таблицы с примерами расписания и регистраций.

### **76. Таймер для готовки**

Функции — установка одного или нескольких таймеров, уведомление по завершении.

TDD — тесты для проверки логики таймера.

ATDD — сценарии для проверки корректного уведомления.

BDD — сценарий «Given таймер запущен, When время истекает, Then получаю уведомление».

SDD — спецификации с примерами времени и ожиданий.

### **77. Трекер портфеля акций**

Функции — добавление акций, обновление цен, расчёт общей стоимости.

TDD — тесты для расчёта стоимости портфеля.

ATDD — приёмочные сценарии для проверки обновления данных.

BDD — сценарий «Given акция добавлена, When обновляется цена, Then портфель пересчитывается».

SDD — таблицы с примерами акций, ценами и итоговыми суммами.

### **78. Трекер онлайн-пожертвований**

Функции — логирование пожертвований, отображение доноров, генерация отчётов.

TDD — тесты для функций регистрации и отчёта.

ATDD — сценарии для проверки корректности отображения истории доноров.

BDD — сценарий «Given пожертвование зафиксировано, When запрашиваю историю, Then вижу данные доноров».

SDD — таблицы с примерами пожертвований и отчётами.



## **79. Напоминание о встречах**

Функции — установка встречи, уведомление, изменение времени.

TDD — тесты для функций установки и редактирования встреч.

ATDD — сценарии для проверки уведомлений.

BDD — сценарий «Given встреча назначена, When наступает время, Then отправляется напоминание».

SDD — спецификации с примерами дат, времени и уведомлений.

## **80. Умный домашний контроллер**

Функции — управление освещением, регулировка температуры, мониторинг состояния устройств.

TDD — тесты для функций управления устройствами.

ATDD — сценарии для проверки корректного переключения состояний.

BDD — сценарий «Given устройство подключено, When изменяю настройки, Then устройство реагирует».

SDD — спецификации с примерами состояний и ожидаемых результатов.

## **81. Система отзывов о продуктах**

Функции — добавление отзыва, редактирование, удаление, фильтрация.

TDD — тесты для операций с отзывами.

ATDD — сценарии для проверки корректности отображения отзывов.

BDD — сценарий «Given отзыв добавлен, When ищу по ключевому слову, Then нахожу нужный отзыв».

SDD — таблицы с примерами отзывов и фильтров.

## **82. Онлайн-рынок (маркетплейс)**

Функции — размещение объявления, поиск, общение с продавцом, отслеживание доставки.

TDD — тесты для функций размещения и поиска.

ATDD — сценарии для проверки коммуникации между продавцом и покупателем.

BDD — сценарий «Given товар размещён, When ищу по названию, Then нахожу объявление».

SDD — спецификации с примерами объявлений и процессом сделки.

## **83. Локатор мероприятий**

Функции — отображение событий, поиск по дате/локации, фильтрация.

TDD — тесты для поиска и фильтрации событий.

ATDD — приёмочные сценарии для проверки корректного отображения мероприятий.

BDD — сценарий «Given список событий, When выбираю дату, Then отображаются мероприятия на эту дату».

SDD — таблицы с датами, локациями и примерами мероприятий.

#### **84. Трекер фитнес-целей**

Функции — установка цели, обновление прогресса, уведомление о достижении.

TDD — тесты для расчёта прогресса и контроля целей.

ATDD — сценарии для проверки уведомлений и обновлений.

BDD — сценарий «Given цель установлена, When достигаю 100%, Then получаю уведомление о достижении».

SDD — спецификации с примерами целей и расчетов прогресса.

#### **85. Приложение для флэш-карт (изучение языков)**

Функции — создание карточек, запуск сессии, отслеживание результатов.

TDD — тесты для создания и обновления карточек.

ATDD — сценарии для проверки эффективности обучения.

BDD — сценарий «Given карточки созданы, When прохожу сессию, Then вижу статистику».

SDD — таблицы с примерами карточек и результатами тестирования.

#### **86. Игровой менеджер задач (с геймификацией)**

Функции — добавление задачи, присвоение баллов, отображение таблицы лидеров.

TDD — тесты для расчёта баллов и обновления таблицы.

ATDD — сценарии для проверки игрового процесса.

BDD — сценарий «Given задача выполнена, When начисляются баллы, Then лидерборд обновляется».

SDD — таблицы с примерами задач, баллов и итоговых позиций.

#### **87. Менеджер кодовых фрагментов**

Функции — сохранение фрагментов кода, редактирование, поиск по тегам.

TDD — тесты для CRUD-операций с кодовыми фрагментами.

ATDD — сценарии для проверки корректности поиска.

BDD — сценарий «Given фрагмент кода сохранён, When ищу по тегу, Then фрагмент находится».

SDD — спецификации с примерами фрагментов и тегов.

### **88. Трекер здоровья**

Функции — логирование симптомов, отображение истории, генерация отчётов.

TDD — тесты для ввода и анализа симптомов.

ATDD — сценарии для проверки корректности отчётов.

BDD — сценарий «Given симптом записан, When запрашиваю историю, Then данные отображаются корректно».

SDD — таблицы с примерами симптомов и аналитическими данными.

### **89. Цифровая библиотека**

Функции — добавление е-книг, поиск, отметка прочтения.

TDD — тесты для функций управления книгами.

ATDD — приёмочные сценарии для проверки корректного отображения библиотеки.

BDD — сценарий «Given книга добавлена, When ищу по названию, Then книга находится».

SDD — таблицы с примерами книг и критериями поиска.

### **90. Менеджер личных финансов**

Функции — учёт доходов и расходов, генерация отчётов, анализ финансов.

TDD — тесты для расчёта баланса и отчётов.

ATDD — сценарии для проверки корректности финансового отчёта.

BDD — сценарий «Given доходы и расходы внесены, When генерирую отчёт, Then баланс рассчитывается корректно».

SDD — таблицы с примерами доходов, расходов и итоговых расчётов.

### **91. Ежедневный планировщик**

Функции — планирование задач на день, установка напоминаний, отображение повестки.

TDD — тесты для функций добавления и редактирования событий.

ATDD — приёмочные сценарии для проверки корректности отображения повестки.

BDD — сценарий «Given задачи запланированы, When просматриваю план, Then все задачи отображаются».

SDD — спецификации с примерами задач и временными промежутками.

## **92. Виртуальный питомец**

Функции — кормление, игра, мониторинг состояния питомца.

TDD — тесты для обновления состояния питомца.

ATDD — сценарии для проверки интерактивности приложения.

BDD — сценарий «Given питомец создан, When кормлю его, Then его состояние улучшается».

SDD — таблицы с примерами взаимодействий и итоговыми состояниями.

## **93. Интеллектуальный To-Do List**

Функции — автоматические рекомендации задач, приоритетизация, отметка выполнения.

TDD — тесты для логики рекомендаций и сортировки задач.

ATDD — сценарии для проверки корректного обновления списка задач.

BDD — сценарий «Given задачи добавлены, When система предлагает приоритеты, Then рекомендации корректны».

SDD — таблицы с примерами задач, приоритетов и итоговых рекомендаций.

## **94. Менеджер закладок URL**

Функции — добавление закладки, организация по категориям, поиск.

TDD — тесты для операций управления закладками.

ATDD — сценарии для проверки удобства поиска и фильтрации.

BDD — сценарий «Given закладка добавлена, When ищу по названию, Then закладка находится».

SDD — таблицы с примерами URL, категорий и ожидаемых результатов.

## **95. Мониторинг IoT устройств**

Функции — добавление устройства, мониторинг состояния, оповещения при сбое.

TDD — тесты для функций контроля и оповещения.

ATDD — сценарии для проверки корректности оповещений при изменении состояния.

BDD — сценарий «Given устройство подключено, When выходит из строя, Then отправляется оповещение».

SDD — спецификации с примерами состояний устройств и ожидаемых уведомлений.

## **96. Менеджер зависимостей задач**

Функции — добавление задач, установка зависимостей между ними, генерация расписания.

TDD — тесты для проверки корректного формирования зависимостей.

ATDD — приёмочные сценарии для проверки логики планирования.

BDD — сценарий «Given задачи с зависимостями добавлены, When строится расписание, Then зависимости учитываются».

SDD — таблицы с примерами зависимостей и итоговыми расписаниями.

## **97. Система рекомендаций рецептов**

Функции — ввод имеющихся ингредиентов, поиск рецептов, предложение вариантов.

TDD — тесты для функций поиска и сопоставления ингредиентов.

ATDD — сценарии для проверки корректности рекомендаций.

BDD — сценарий «Given у меня есть ингредиенты, When запрашиваю рецепты, Then получаю подходящие варианты».

SDD — таблицы с ингредиентами и соответствующими рецептами.

## **98. Система сопоставления пожертвований**

Функции — регистрация доноров, сопоставление пожертвований, генерация отчётов.

TDD — тесты для функций сопоставления и отчётности.

ATDD — сценарии для проверки корректности подбора доноров и сумм.

BDD — сценарий «Given донор зарегистрирован, When делается пожертвование, Then система сопоставляет и выдаёт отчёт».

SDD — таблицы с примерами доноров, сумм и сопоставлений.

## **99. Трекер потребления энергии**

Функции — логирование потребления, расчёт стоимости, генерация аналитических отчётов.

TDD — тесты для функций расчёта затрат на основе потребления.

ATDD — сценарии для проверки корректности отчётов по энергопотреблению.

BDD — сценарий «Given данные потребления внесены, When генерирую отчёт, Then рассчитывается стоимость».

SDD — таблицы с примерами данных потребления и итоговыми расчетами.

## **100. Доска объявлений сообщества**

Функции — публикация объявлений, комментирование, модерация контента.

TDD — тесты для операций публикации, редактирования и удаления объявлений.

ATDD — приёмочные сценарии для проверки корректного отображения и модерации.

BDD — сценарий «Given объявление опубликовано, When добавляю комментарий, Then комментарий отображается».

SDD — спецификации с таблицами объявлений, комментариев и правил модерации.

Каждое задание можно детально проработать:

С использованием TDD — написание юнит-тестов для каждой отдельной функции, реализация минимально рабочего кода и последующий рефакторинг.

С использованием ATDD — определение приёмочных критериев с участием всех заинтересованных сторон и автоматизация соответствующих сценариев.

С использованием BDD — создание сценариев в формате Given–When–Then (например, на языке Gherkin), которые служат и документацией, и автоматизированными тестами.

С использованием SDD — сбор конкретных примеров и составление спецификаций, преобразованных в автоматизированные тесты.

Ниже приведён вариант задания «Менеджер подписок», а также подробная реализация этого примера с использованием подходов TDD, ATDD, BDD и SDD.

### **1.3.1. Пример. Вариант задания: менеджер подписок**

Цель работы разработать приложение, которое позволяет пользователю:

- 1) добавлять подписки (например, на стриминговые сервисы, журналы, софт и т.д.) с указанием названия, стоимости, периода оплаты (ежемесячно, ежегодно и т.д.) и даты следующего платежа;
- 2) просматривать список активных подписок;
- 3) редактировать и удалять подписки;
- 4) получать уведомление (в виде сообщения) за определённое количество дней до даты оплаты.

**Этап №1.** Реализация с помощью TDD (Test-Driven Development). Были написаны юнит-тесты для ключевых функций менеджера подписок, реализован минимально необходимый код и провести рефакторинг.

Ниже на рис. 14 представлен пример реализации тестов на языке Python (с использованием unittest):

```
import unittest
from datetime import datetime, timedelta
from subscription_manager import SubscriptionManager, Subscription

class TestSubscriptionManager(unittest.TestCase):

    def setUp(self):
        self.manager = SubscriptionManager()
        # Добавим тестовую подписку
        self.subscription = Subscription(
            name="Netflix",
            cost=9.99,
            period="monthly",
            next_payment_date=datetime.now() + timedelta(days=5) # через 5 дней
        )

    def test_add_subscription(self):
        self.manager.add_subscription(self.subscription)
        self.assertIn(self.subscription, self.manager.subscriptions)

    def test_remove_subscription(self):
        self.manager.add_subscription(self.subscription)
        self.manager.remove_subscription(self.subscription.name)
        self.assertNotIn(self.subscription, self.manager.subscriptions)

    def test_get_upcoming_notifications(self):
        # Установим уведомление за 7 дней до платежа
        self.manager.add_subscription(self.subscription)
        # Если уведомление настроено на 7 дней, то подписка с платежом через 5 дней
        notifications = self.manager.get_upcoming_notifications(days_before=7)
        self.assertIn(self.subscription, notifications)

    def test_edit_subscription(self):
        self.manager.add_subscription(self.subscription)
        new_cost = 12.99
        self.manager.edit_subscription(self.subscription.name, cost=new_cost)
        edited_subscription = self.manager.get_subscription(self.subscription.name)
        self.assertEqual(edited_subscription.cost, new_cost)

if __name__ == "__main__":
    unittest.main()
```

Рисунок 12. Пример написания юнит-тесты для функций менеджера подписок

В соответствии с методологией TTD на втором этапе был реализован код, пример которого представлен на рис. 15.

```
from datetime import datetime

class Subscription:
    def __init__(self, name, cost, period, next_payment_date):
        self.name = name
        self.cost = cost
        self.period = period # например, "monthly", "yearly" и т.д.
        self.next_payment_date = next_payment_date

class SubscriptionManager:
    def __init__(self):
        self.subscriptions = []

    def add_subscription(self, subscription):
        self.subscriptions.append(subscription)

    def remove_subscription(self, name):
        self.subscriptions = [s for s in self.subscriptions if s.name != name]

    def edit_subscription(self, name, **kwargs):
        subscription = self.get_subscription(name)
        if subscription:
            for key, value in kwargs.items():
                if hasattr(subscription, key):
                    setattr(subscription, key, value)

    def get_subscription(self, name):
        for s in self.subscriptions:
            if s.name == name:
                return s
        return None

    def get_upcoming_notifications(self, days_before):
        upcoming = []
        now = datetime.now()
        for s in self.subscriptions:
            if (s.next_payment_date - now).days <= days_before:
                upcoming.append(s)
        return upcoming
```

*Рисунок 13. Реализация функций менеджера подписок*

Запустив файл с тестами (например, `python -m unittest test_subscription_manager.py`), можно убедиться, что все они успешно пройдены. При необходимости возможно выполнить рефакторинг кода.

**Этап №2.** Реализация с помощью ATDD (Acceptance Test-Driven Development). Реализованы приёмочные тесты для сценариев использования приложения, которые были согласованы с конечными пользователями.



### Сценарий 1. Добавление подписки.

Предусловие: пользователь заходит в раздел «Добавить подписку».

Действия:

- 1) пользователь вводит название подписки (например, «Netflix»);
- 2) указывает стоимость (например, 9.99);
- 3) выбирает период оплаты («monthly»);
- 4) устанавливает дату следующего платежа (например, через 5 дней от текущей даты);
- 5) нажимает кнопку «Сохранить».

Ожидаемый результат: подписка «Netflix» появляется в списке активных подписок.

### Сценарий 2. Уведомление о предстоящем платеже.

Предусловие: пользователь имеет активную подписку с датой следующего платежа через 5 дней.

Действия:

- 1) пользователь настраивает уведомления за 7 дней до даты платежа;
- 2) приложение проверяет даты и отправляет уведомление.

Ожидаемый результат: подписка «Netflix» отображается в списке подписок, для которых нужно отправить уведомление.

**Примечание:** Отчёт так же должен содержать приёмочные тесты, удачную их проверку и код. Пример реализации можно найти в теоретической части практического задания.

**Этап №3.** Реализация с помощью BDD (Behavior-Driven Development). Созданы сценарии на понятном для всех участников проекта языке (Gherkin), которые описывают поведение системы.

Пример сценария на языке Gherkin представлен на рис. 16.

```
Feature: Управление подписками
  Чтобы не забывать оплачивать услуги
  Как пользователь менеджера подписок
  Я хочу добавлять, редактировать, удалять подписки и получать уведомления о предстоящих платежах

Scenario: Добавление новой подписки
  Given я нахожусь на странице "Добавить подписку"
  When я ввожу название "Netflix", стоимость "9.99", период "monthly" и дату следующего платежа "2025-02-10"
  And я нажимаю кнопку "Сохранить"
  Then подписка "Netflix" появляется в списке активных подписок

Scenario: Получение уведомления о предстоящем платеже
  Given у меня есть подписка "Netflix" с датой платежа через 5 дней
  When я настраиваю уведомления за "7" дней до даты платежа
  Then система должна показать подписку "Netflix" в списке уведомлений
```

Рисунок 14. Пример сценария на языке Gherkin

**Примечание:** Реализацию сценариев необходимо автоматизировать, например, с помощью таких инструментов, как Behave (для Python) или Cucumber (для других языков). Данные действия должны быть отражены в итоговом отчёте. Пример автоматизации сценарием можно найти в теоретической части практического задания.

**Этап №4.** Реализация с помощью SDD (Specification by Example). В проекте созданы спецификации требований с использованием конкретных примеров, которые затем были преобразованы в автоматизированные тесты.

Спецификации для подхода SDD описаны в виде табл. 2.

Таблица 2. Пример описания спецификаций требований

Название подписки	Стоимость	Период	Дата следующего платежа	Ожидаемое уведомление при настройке на X дней
Netflix	9.99	monthly	2025-02-10	Если $X = 7$ , уведомление должно появиться, так как $(2025-02-10 - \text{текущая дата}) \leq 7$ дней
Spotify	4.99	monthly	2025-02-20	Если $X = 7$ , уведомление не появится, так как $(2025-02-20 - \text{текущая дата}) > 7$ дней

Для каждой подписки задаются конкретные входные данные: название, стоимость, период и дата следующего платежа.

Спецификация описывает, при каком значении параметра «уведомления за X дней» подписка должна попасть в список предстоящих уведомлений.

**Примечание:** Отчёт так же должен содержать автоматизированные тесты, которые создаются на основе приведённых таблиц, что обеспечит соответствие реализации ожиданиям, зафиксированным в спецификации. Пример реализации можно найти в теоретической части практического задания.

Вывод: в данном примере «Менеджер подписок» показана реализация приложения с использованием четырёх подходов разработки через тестирование:

1. TDD — Написаны юнит-тесты, реализована базовая функциональность и выполнен рефакторинг кода.
2. ATDD — Определены приёмочные тесты, согласованные с участниками процесса, которые описывают ключевые сценарии использования.
3. BDD — Сформулированы сценарии на языке Gherkin, понятные как разработчикам, так и бизнес-пользователям.
4. SDD — Составлена спецификация с конкретными примерами, позволяющая создать живую документацию и автоматизированные тесты на её основе.

Эта комплексная реализация позволяет не только протестировать функциональность менеджера подписок, но и продемонстрировать интеграцию различных подходов к разработке через тестирование, что повышает качество и надёжность конечного продукта.

#### **1.4. Итоговый отчёт**

По результатам индивидуальной работы отчёт должен содержать:

- 1) Титульный лист, включающий в себя наименование работы, автора, дату выполнения, название учебной дисциплины.
- 2) Документирование каждого этапа работы.
- 3) Описание теоретических основ, практических примеров, результатов тестирования и анализа.
- 4) Заключение, включающее общую оценку качества программного продукта и документации, выводы о проделанной работе.
- 5) Приложения (при необходимости): дополнительные материалы (коды, логи тестирования, скриншоты и т. д.).

Отчёт должен быть оформлен в соответствии с ГОСТ 19.401-78 и ГОСТ 34.602-2020. Требования включают стандарты на титульный лист, использование единообразного шрифта, оформление таблиц и диаграмм, наличие нумерации страниц.

Итоговая оценка работы будет зависеть от полноты отчёта, качества выполнения задания, соответствия оформления стандартам и презентации результатов на защите.