Федеральное государственное бюджетное образовательное учреждение высшего образования
«Казанский национальный исследовательский технологический университет» (ФГБОУ ВО «КНИТУ»)

Институт управления, автоматизации и информационных технологий
Кафедра «Интеллектуальных систем и управления информационными ресурсами»

Курсовой проект по дисциплине «Разработка пользовательского интерфейса»

на тему: «Разработка клиентской части для инструмента анализа покрытия автоматизированными тестами»

Выполнил: студент группы 4311-21 Козлов И.В.

Проверил: доцент кафедры ИСУИР, к.ф.-м.н. Мангушева А.Р.

Цель и задачи курсового проекта

Целью курсового проекта является создание веб-интерфейса для инструмента, специально адаптированного под анализ покрытия автоматизированными тестами, который будет интуитивно понятен и функционален.

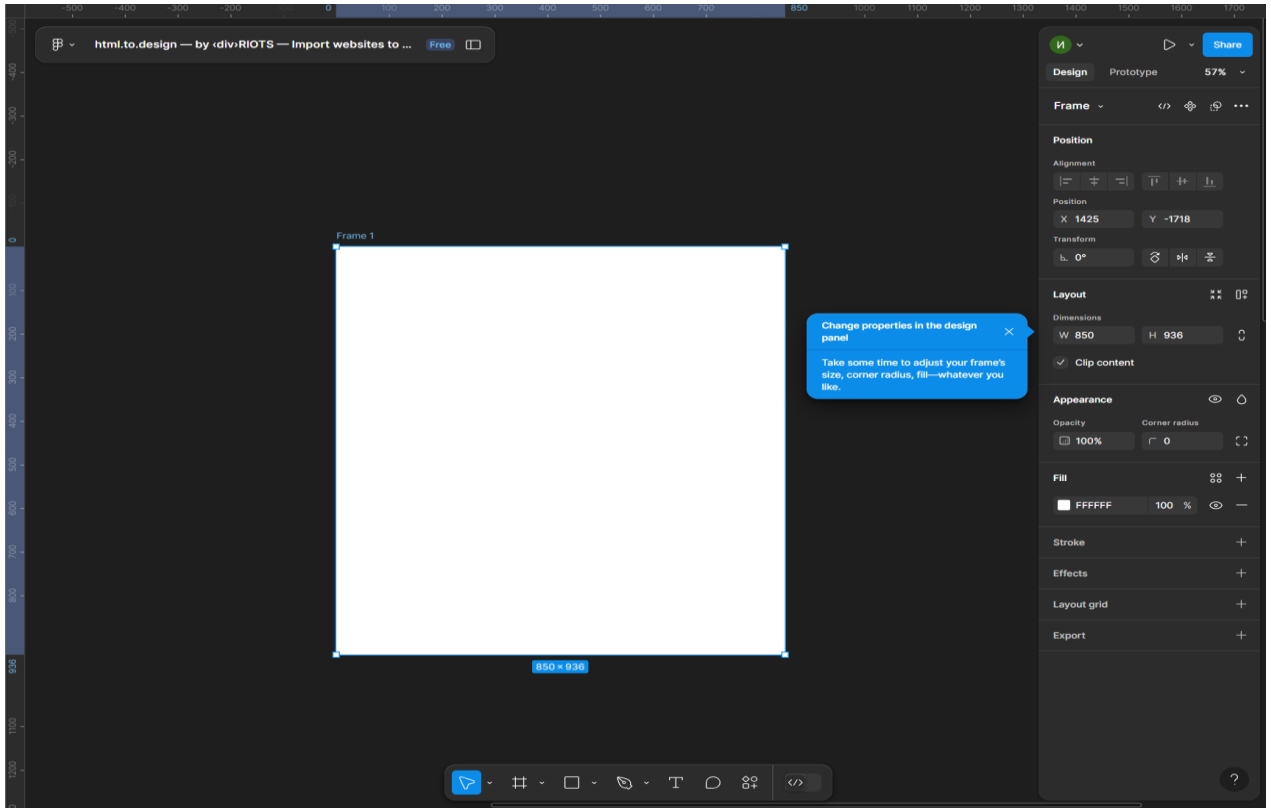
Для достижения поставленной цели выдвигаются следующие задачи:
1. Сформулировать требования, выдвигаемые к продукту.
2. Разработать макет клиентской части, отвечающий сформулированным функциональным требованиям.
3. Создать визуальную структуру сайта, используя простые языки разметки.
4. Внедрить интерактивность с жестко прописанными в программном коде данными.
5. Добавить связку клиентской части с серверной.

Анализ требований

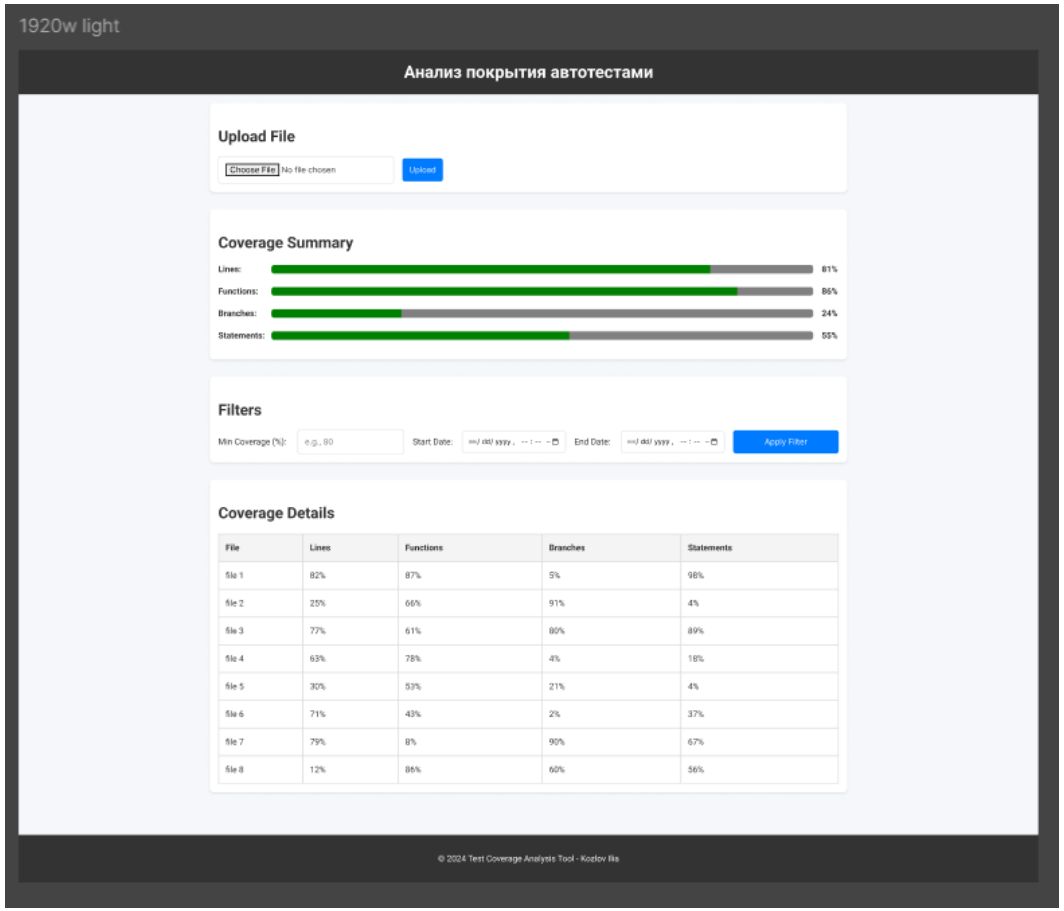
Функциональные	Технические
Приложение представляет сводную статистику по покрытию автоматизированными тестами	Используемые технологии
Статистика по каждому запуску	Загруженные данные должны сохраняться на сервере через API-эндпоинт
Полная информация о файле	Структура данных файла

Макет интерфейса

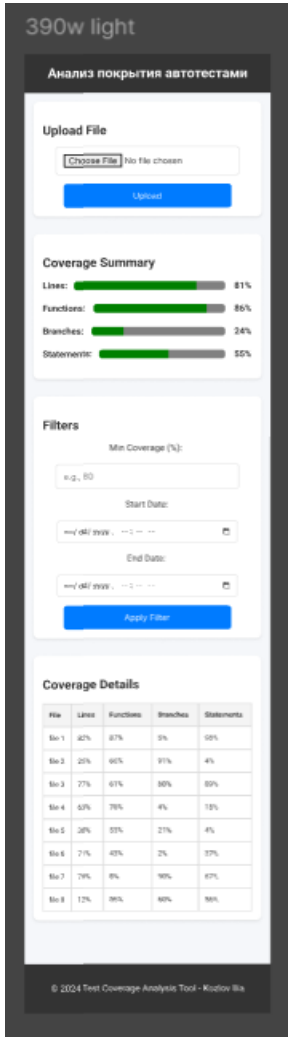
Для проектирования макета выбран графический редактор Figma



Макет сайта для персонального компьютера



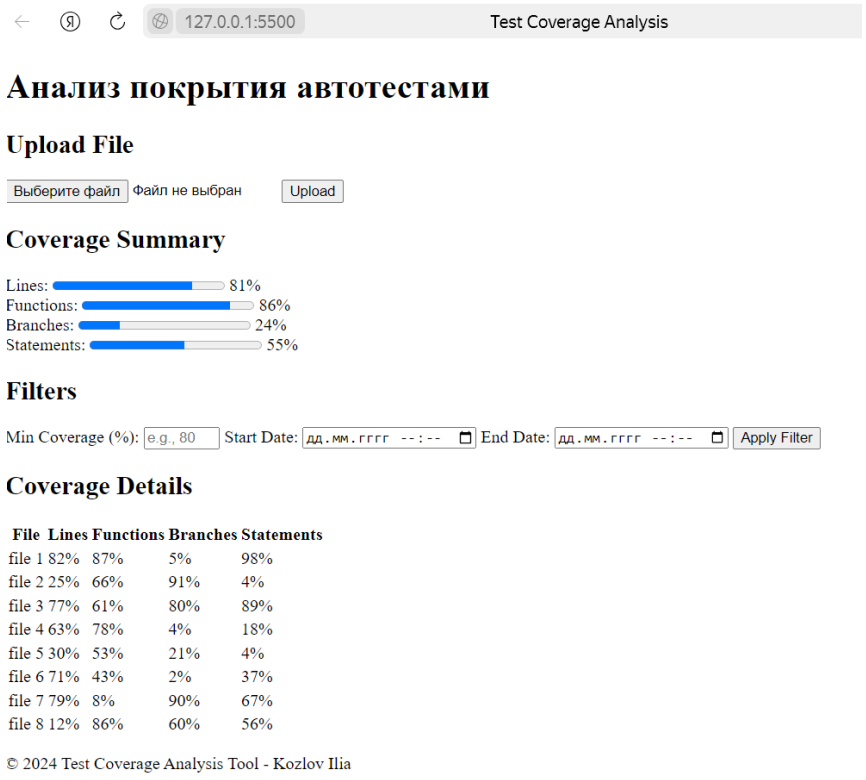
Макет сайта для мобильного веба



Верстка сайта по макету

Создание веб-интерфейса на HTML

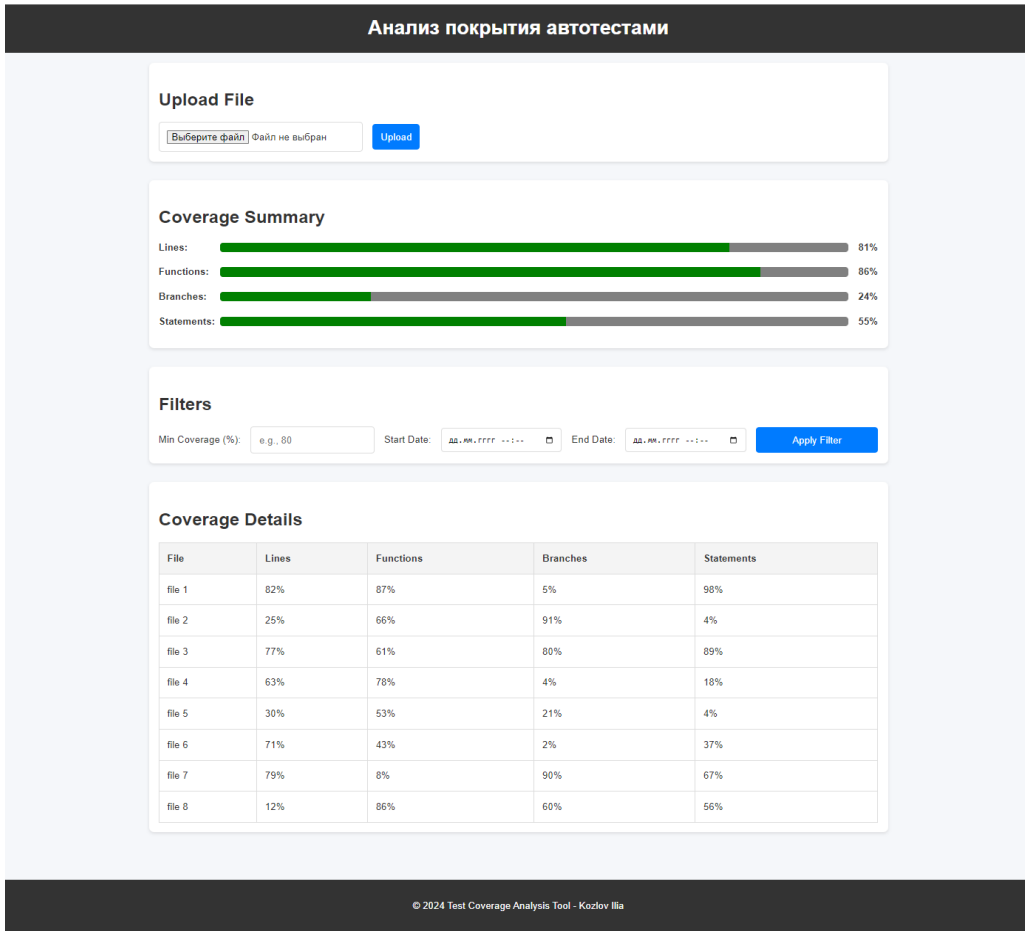
```html
<body>
  <header id="app-header">
    <h1>Анализ покрытия автотестами</h1>
  </header>
  <main>
    <section id="file-upload">
    <h2>Upload File</h2>
    <form id="file-upload-form">
      <input type="file" id="file-input" accept="application/json">
      <button type="submit">Upload</button>
    </form>
  </section>
    <section id="coverage-summary">
      <h2>Coverage Summary</h2>
      <div class="stats-bar">
        <label>Lines:</label>
        <progress id="lines-coverage" max="100" value="81"></progress>
        <span id="lines-percentage">81%</span>
      </div>
      <div class="stats-bar">
        <label>Functions:</label>
        <progress id="functions-coverage" max="100" value="86"></progress>
        <span id="functions-percentage">86%</span>
      </div>
      <div class="stats-bar">
        <label>Branches:</label>
        <progress id="branches-coverage" max="100" value="24"></progress>
        <span id="branches-percentage">24%</span>
      </div>
      <div class="stats-bar">
        <label>Statements:</label>
        <progress id="statements-coverage" max="100" value="55"></progress>
        <span id="statements-percentage">55%</span>
      </div>
    </section>
    <section id="filters-container">
      <h2>Filters</h2>
      <form id="filter-form">
        <label for="min-coverage">Min Coverage (%):</label>
        <input type="number" id="min-coverage" name="min-coverage" min="0" max=
```

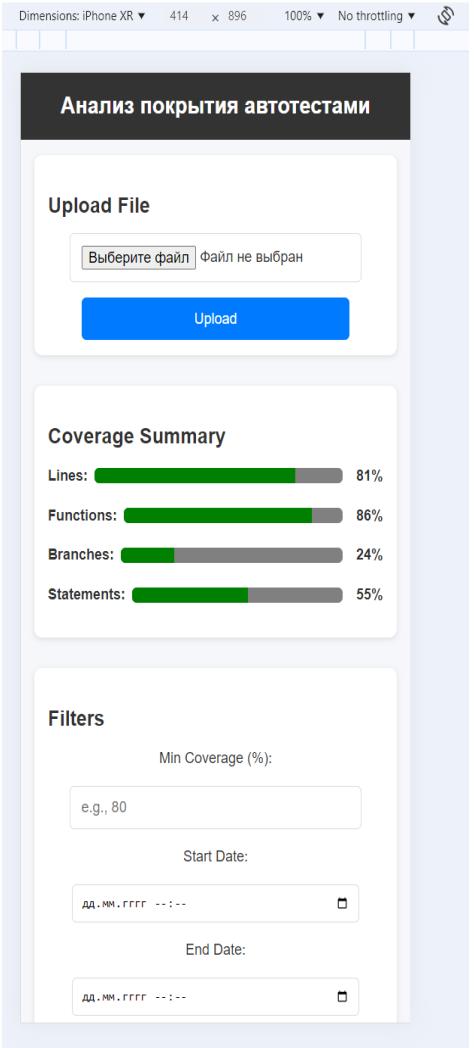


Верстка сайта по макету

Стилизация элементов

```
148
149   .modal.hidden {
150     display: none;
151   }
152
153   .modal-content {
154     background: #fff;
155     padding: 2em;
156     border-radius: 10px;
157     width: 90%;
158     max-width: 500px;
159     box-shadow: 0 5px 15px rgba(0, 0, 0, 0.3);
160     position: relative;
161   }
162
163   .close-btn {
164     position: absolute;
165     top: 10px;
166     right: 15px;
167     font-size: 1.5em;
168     color: #333;
169     cursor: pointer;
170   }
171
172   .close-btn:hover {
173     color: #ff0000;
174   }
175
176   .modal button {
177     padding: 0.8em;
178     font-size: 1em;
179     border: 1px solid #ddd;
180     border-radius: 5px;
181   }
182
183   .modal button {
184     background-color: #007bff;
185     color: #fff;
186     cursor: pointer;
187     border: none;
188   }
189
190   .modal button:hover {
191     background-color: #0056b3;
192   }
193
194   /* Анимация модального окна */
195   .modal.show {
196     animation: fadeIn 0.3s ease-out forwards;
197   }
```



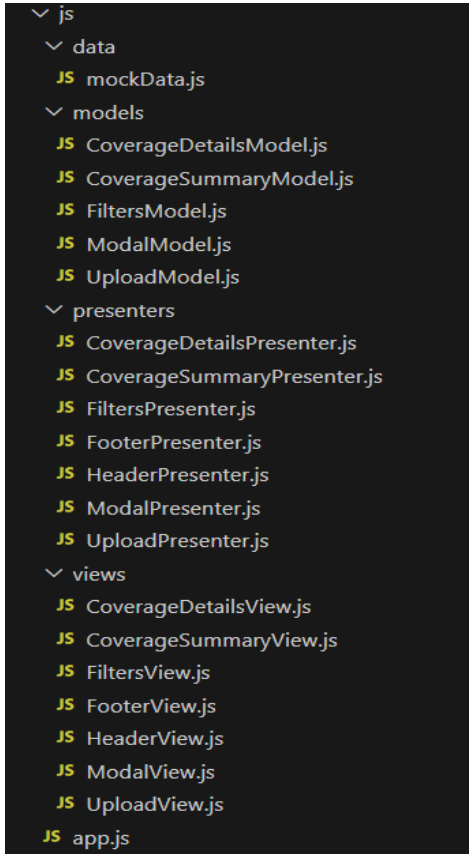


Внедрение интерактивности

- JavaScript
- MVP (Model-View-Presenter)
- Структура файлов и папок
- Применение принципов ООП

```
You, 3 weeks ago | 1 author (You)
export default class UploadModel {
    constructor() {
      this.files = [];
    }

    addFile(file) {
      const newFile = {
        name: file.name,
        size: file.size,
        type: file.type,
        uploadTime: new Date().toISOString(),
      };
      this.files.push(newFile);
      return newFile;
    }
  }
        You, 3 weeks ago • init Test Coverage Ana
```



Пошаговая реализация

Model

```
You, 3 weeks ago | 1 author (You)
export default class UploadModel {
    constructor() {
      this.files = [];
    }

    addFile(file) {
      const newFile = {
        name: file.name,
        size: file.size,
        type: file.type,
        uploadTime: new Date().toISOString(),
      };
      this.files.push(newFile);
      return newFile;
    }
  }
        You, 3 weeks ago • init Test Coverage Ana
```

View

```
You, 3 weeks ago | 1 author (You)
export default class UploadView {
  constructor() {
    this.container = document.getElementById("file-upload");
  }

  render() {
    this.container.innerHTML = `
        <h2>Upload Test Coverage Report</h2>
        <form id="upload-form">
          <input type="file" id="coverage-file" accept=".json" />
          <button type="submit">Upload</button>
        </form>
      `;
    this.uploadForm = document.getElementById("upload-form");
    this.fileInput = document.getElementById("coverage-file");
  }

  bindFileUpload(handler) {
    this.uploadForm.addEventListener("submit", (event) => {
      event.preventDefault();
      const file = this.fileInput.files[0];
      if (file) {
        handler(file);
        this.fileInput.value = ""; // Сброс input после загрузки
      }
    });
  }
}
```

Пошаговая реализация

Presenter

```
You, 1 second ago | 1 author (You)
export default class UploadPresenter {
    constructor(model, view) {
      this.model = model;
      this.view = view;

      this.view.render();
      this.view.bindFileUpload(this.handleFileUpload.bind(this));
    }

    handleFileUpload(file) {
      const newFile = this.model.addFile(file);
      console.log("Uploaded file:", newFile);
    }
  }
      You, 3 weeks ago • init Test Coverage Analysis Tool
```

Инициализация классов

```
// --- Upload ---
const uploadModel = new UploadModel();
const uploadView = new UploadView();
const uploadPresenter = new UploadPresenter(uploadModel, uploadView);
```

Связь с серверной частью

Пример потока данных

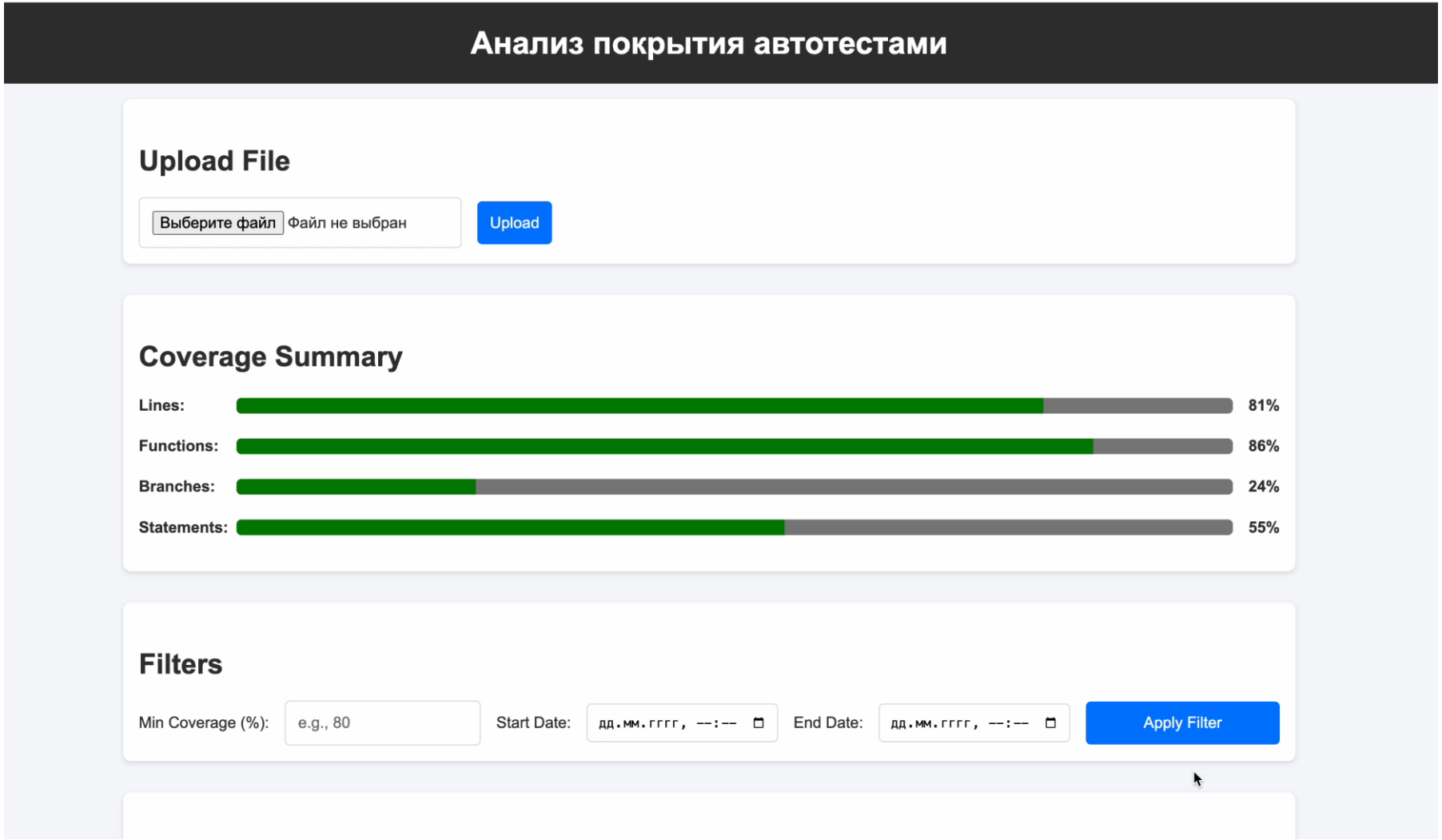
1. Пользователь загружает файл через форму.

2. Презентер вызывает метод модели для отправки данных на сервер через POST-запрос.

3. Сервер добавляет новые данные и возвращает результат.

4. Приложение получает обновленные данные и рендерит их на странице.

Видеодемонстрация работы



Ссылки на материалы

Развернутая клиентская часть	https://xytyfresch.github.io/kursach/
Репозиторий с кодом	https://github.com/xytyfresch/kursach

Заключение

- Во время выполнения курсового проекта были проанализированы функциональные и технические требования.
- С помощью инструмента Figma, опираясь на ранее составленные требования был спроектирован макет будущего приложения.
- Статическая версия клиентской части была написана с использованием HTML и CSS.
- Проект был переведен на архитектуру MVP.
- Успешно реализовано клиент-серверное взаимодействие.