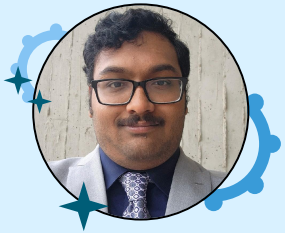
Lighting Talk:
User Testing

By: sdmay26-16

Team Members

Onur Onal

Project Lead/Project Manager

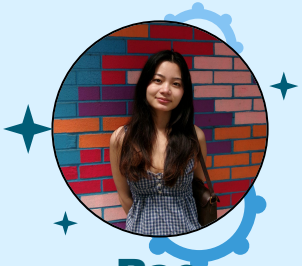


Prakeerth

Frontend & Scheduler/Organizer

Erroll

Lead Frontend UX/UI designer



Bae

Backend

Kemal

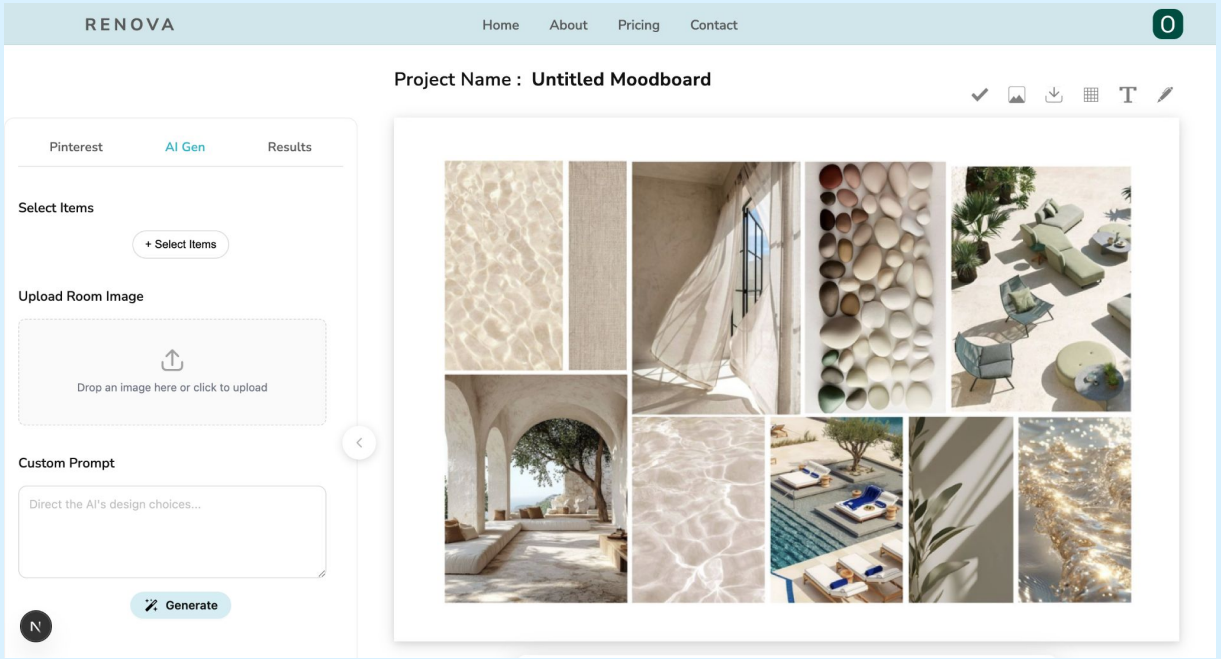
Fullstack

Om

Backend

RENOVA

- **AI Moodboards for interior design**
- **Edit and renovate rooms visually**
- **Blend creativity with AI generation**



RENOVA

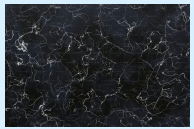
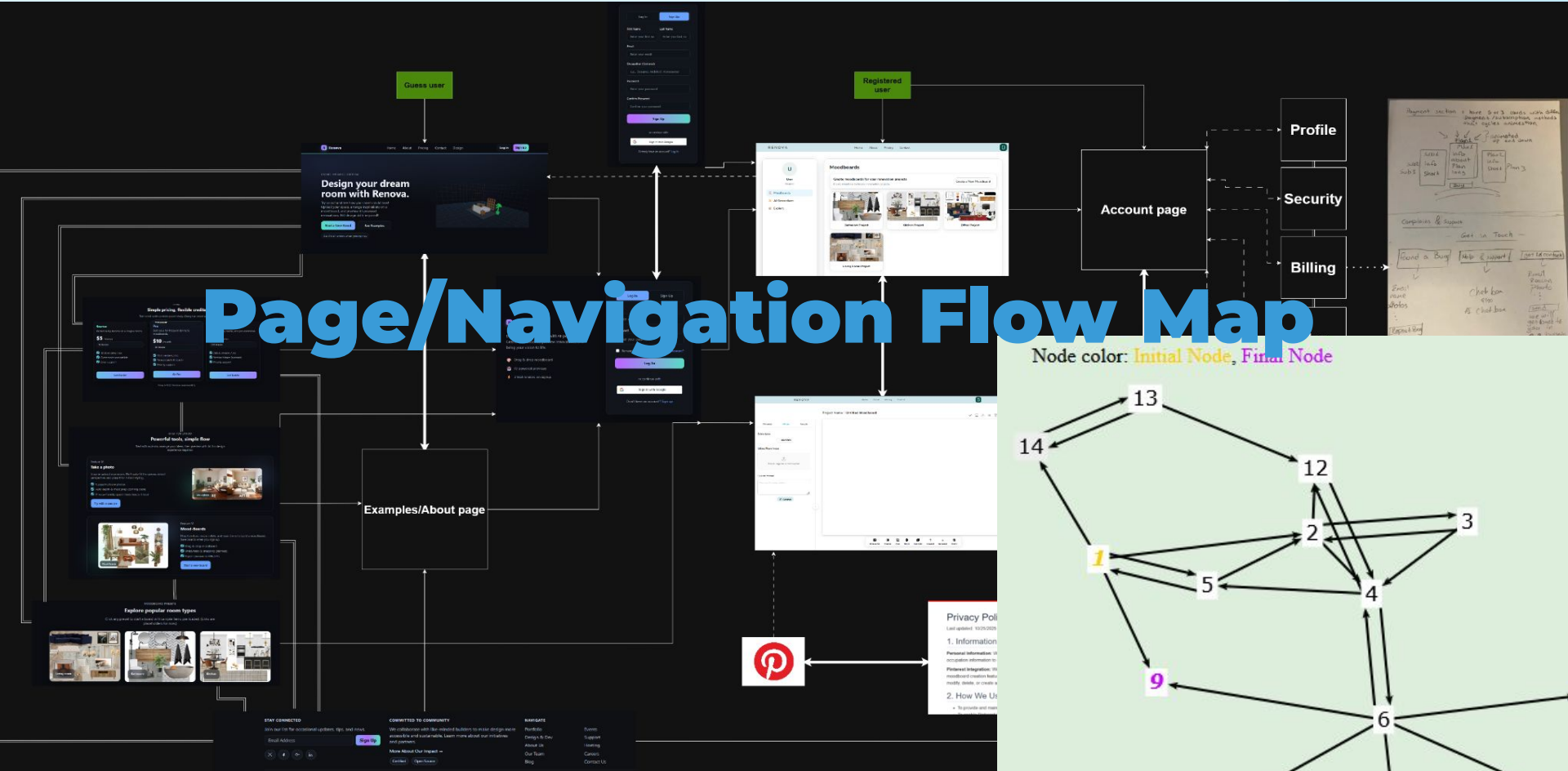


TABLE OF CONTENTS

Testing Philosophy

- Test how everything works together
- Find problems early
- Keep system stable with auto tests

Node Mapping Coverage



Page/Navigation Flow Map

Node Mapping Coverage

Node

1. Landing
2. Login
3. Signup
4. Dashboard
5. Example/About
6. Account
7. Plan
8. Billing
9. Support
10. Profile
11. Security
12. New board
13. Pinterest
14. Privacy

Edge

1. Landing → Login
2. Login/Signup → Dashboard
3. Login ↔ Signup
4. Landing ↔ Example/About
5. Dashboard ↔ Account
6. Account → Plan
7. Account → Billing
8. Account → Support
9. Account → Profile
10. Account → Security
11. Dashboard → Examples/About
12. Dashboard ↔ New board
13. Example→Login→New board
14. Account → Dashboard
15. Landing→ Support
16. Pinterest ↔ Privacy
17. Pinterest → New board

350 requirements are needed for **Simple Paths**
109 requirements are needed for **Prime Paths**
94 test paths are needed for **Prime Path Coverage**

Frontend / E2E Testing

- Overall uses Jest + React Testing Library
- Testing React components and UI behavior
- Cypress for End-to-End Testing
- Login/Signup Authflow Testing

```
it('loads user after successful auth check', async () => {
  const mockUser = { email: 'test@example.com' };
  authAPI.getCurrentUser.mockResolvedValue(mockUser);
  localStorage.setItem('accessToken', 'test-token');

  await act(async () => {
    render(
      <AuthProvider>
        <TestComponent />
      </AuthProvider>
    );
  });

  // Wait for state to settle
  await screen.findByTestId('user');

  expect(screen.getByTestId('user')).toHaveTextContent('test@example.com');
  expect(screen.getByTestId('loading')).toHaveTextContent('ready');
});
```

Test case inside of AuthContext.unit.test.jsx

Backend Testing

Jest for Unit Testing

- Tested logic and controllers with mocked Database + auth helpers
- Verified validation, responses, and error handling

Supertest + MySQL Test container for Integration Testing

- Full auth, user, and moodboard flows
- CRUD operations, permissions, and end-to-end API behavior

Backend Health Endpoint

Prevent User Frustration

Maintain Trust During Updates

Enable Fast Recovery

http://localhost:8080/health

Pretty-print


```
{
  "status": "ok",
  "timestamp": "2025-11-10T08:47:51.918Z",
  "uptime": 40844,
  "environment": "development",
  "memory": {
    "rss": 60,
    "heapUsed": 9,
    "heapTotal": 10
  },
  "services": {
    "database": {
      "status": "ok",
      "responseTime": "100ms"
    }
  },
  "responseTime": "113ms"
}
```


http://localhost:8080/health/ready

Pretty-print


```
{
  "status": "ready",
  "timestamp": "2025-11-10T08:49:15.806Z"
}
```


http://localhost:8080/health/live

Pretty-print


```
{
  "status": "ok",
  "timestamp": "2025-11-10T08:49:38.157Z"
}
```

THANKS!

Does anyone have any questions?