**MessFinder Project Development Plan**

**1. Project Overview**

* **Title**: MessFinder: Connecting Owners and Tenants for Budget Stays
* **Problem Statement**: Students and professionals in India struggle to find affordable paying guest or shared room accommodations. Owners face challenges advertising vacant rooms to the right audience.
* **Solution**: A web-based platform where owners post room listings (location, amenities, food options, price) and users search for rooms using dropdown-based location filters (State, District, City/Area, PIN code).
* **Objectives**:
  1. Build a web app with owner and user interfaces.
  2. Enable owners to post/edit listings with location, room details, and photos.
  3. Allow users to search rooms by location and contact owners.
  4. Deliver a functional prototype, report, and presentation within 1 month.
* **Scope**:
  1. **Include**: Owner dashboard (post/edit listings), user search (dropdown-based), Firebase authentication, photo uploads, contact system.
  2. **Exclude**: Geolocation/maps, payment integration, mobile app, admin panel.
* **Target Users**: Owners (landlords), users (students/professionals).

**2. Requirements (Refined)**

**Functional Requirements**

* **Owner Section**:
  + Register/login (Firebase Authentication, email/password).
  + Post/edit/delete listings:
    - Location: State, District, City/Area, PIN code (dropdowns).
    - Room details: Beds, desks, capacity (e.g., 3-person).
    - Amenities: Wi-Fi, water, etc.
    - Food: Available (with price), included, or not available.
    - Price: Monthly rent per person.
    - Photos: Upload 1–5 images (Firebase Storage).
  + View user inquiries (via email or in-app messaging).
* **User Section**:
  + Browse as guest or register/login.
  + Search rooms via dropdowns (State, District, City/Area, PIN) and filters (price, room type, food).
  + View listing details (photos, amenities, price, owner contact).
  + Contact owner (email or in-app message).
* **Core Features**:
  + Firebase Authentication for secure login.
  + Firestore for storing listings, users, and inquiries.
  + Firebase Storage for photo uploads.
  + Simple messaging or contact system.

**Non-Functional Requirements**

* **Performance**: Search results in < 1 second for 100 listings.
* **Usability**: Intuitive, mobile-responsive UI.
* **Security**: Secure authentication, input validation.
* **Scalability**: Handle 500 users and 200 listings initially.

**Academic Requirements**

* **Deliverables**: Working web app, project report (10–15 pages), presentation (10–12 slides), demo, viva.
* **Timeline**: 1 month (May 1–31, 2025).
* **Team**: 9 members (3 developers, 6 non-developers).
* **Tech Stack**: React, Node.js, Express, Firebase (Authentication, Firestore, Storage).

**3. Team Structure and Task Division**

* **Team Size**: 9 members.
  + **Developers (3)**: You (lead), Dev1, Dev2 (coding).
  + **Non-Developers (6)**: Two groups of 4 (Group A, Group B); you oversee non-dev tasks.
* **Rationale**:
  + Developers focus on coding to meet the deadline.
  + Non-developers handle research, testing, and documentation to contribute without technical skills.
  + Removing the admin panel reduces dev workload by ~20%, making the project feasible.
  + Tasks weighted by effort (coding heaviest, documentation moderate, testing lighter).

**Team Roles and Tasks**

1. **Developer Group (3 members)**:
   * **You (Lead Developer)**:
     + **Tasks** (40% of dev work):
       - Back-end: Set up Express server, Firebase integration (Authentication, Firestore, Storage), core APIs (listings, search, inquiries).
       - Project oversight: Coordinate with non-dev groups, ensure timeline adherence.
       - Deployment: Host app on Vercel or Firebase Hosting.
       - Messaging system: Build in-app inquiry or email contact.
     + **Why**: You handle critical back-end and coordination to ensure quality and deadline adherence.
   * **Dev1**:
     + **Tasks** (30% of dev work):
       - Front-end: Owner dashboard (post/edit listing forms, listing table).
       - Photo uploads: Integrate Firebase Storage.
       - UI polish: Ensure responsive design.
     + **Why**: Owner section is moderately complex, suitable for a single developer.
   * **Dev2**:
     + **Tasks** (30% of dev work):
       - Front-end: User section (search page with dropdowns, listing details).
       - Search filters: Implement price, room type, food filters.
       - Testing: Write unit tests for APIs.
     + **Why**: User section is critical for demo; filters and testing balance workload.
   * **Collaboration**: Use GitHub for version control; daily standups (5–10 min) via WhatsApp or Discord to sync.
2. **Non-Developer Group A (4 members)**:
   * **Tasks**:
     + **Location Data Collection** (Week 1): Source India State/District/City/PIN data (e.g., GitHub repos, government sites). Format as JSON for dropdowns.
     + **User Testing** (Weeks 3–4): Test app as owners and users; provide feedback on UI, search, bugs.
     + **Sample Data Creation** (Week 2): Create 20–30 sample listings (e.g., rooms in Karnataka, Delhi).
     + **Presentation Slides** (Week 4): Draft 10–12 slides (problem, solution, demo, tech stack, results).
   * **Why**: Data and testing are practical, non-technical tasks; slides leverage group size.
3. **Non-Developer Group B (4 members)**:
   * **Tasks**:
     + **UI Mockups** (Week 1): Use Figma to create wireframes for owner and user interfaces.
     + **Documentation** (Weeks 3–4): Write project report (10–15 pages: intro, methodology, results, conclusion).
     + **Competitor Analysis** (Week 1): Research platforms like NoBroker, PG Finder for report’s literature review.
     + **Demo Support** (Week 4): Record demo video, prepare viva Q&A.
   * **Why**: Mockups and report suit non-technical skills; analysis adds academic value.
4. **Your Non-Dev Role (You, as Lead)**:
   * **Tasks**:
     + Review Group A/B outputs (data, mockups, report, slides).
     + Write project proposal (1 page, Week 1).
     + Prepare demo script and present it (Week 4).
   * **Why**: Your oversight ensures non-dev work aligns with the prototype.

**Task Weighting**

* **Development (60%)**: You (40%), Dev1 (30%), Dev2 (30%) for balanced coding effort.
* **Non-Development (40%)**:
  + Group A (20%): Data, testing, slides.
  + Group B (15%): Mockups, report, analysis.
  + You (5%): Oversight, demo prep.

**Action**: Confirm task division or adjust roles. Share team member names for a personalized task table.

**4. System Design (Updated)**

**System Architecture**

* **Client-Server Model**:
  + **Client**: React web app for owner and user interfaces.
  + **Server**: Node.js + Express for APIs not handled by Firebase.
  + **Database**: Firestore for listings, users, inquiries.
  + **Storage**: Firebase Storage for photos.
  + **Authentication**: Firebase Authentication.
* **Diagram**:

text

[Owner/User] --> [React Front-end]

|

v

[Node.js/Express] --> [Firebase: Firestore, Storage, Auth]

* **Components**:
  + Front-end: React components for UI.
  + Back-end: Express APIs for search and messaging; Firebase for auth and CRUD.
  + Firebase: Manages data, photos, security.

**Tech Stack**

* **Front-end**: React.js.
* **Back-end**: Node.js + Express.
* **Database**: Firebase Firestore.
* **Storage**: Firebase Storage.
* **Authentication**: Firebase Authentication.
* **Hosting**: Vercel or Firebase Hosting.
* **Tools**: Git/GitHub, Figma, Postman, VS Code.

**Database Schema (Firestore)**

* **Users Collection**:

json

{

"uid": "user123",

"email": "user@example.com",

"role": "owner" | "user",

"name": "John Doe",

"phone": "1234567890"

}

* **Listings Collection**:

json

{

"id": "listing456",

"owner\_id": "user123",

"location": {

"state": "Karnataka",

"district": "Bangalore Urban",

"city\_area": "Koramangala",

"pin": "560034"

},

"room\_details": {

"beds": 3,

"desks": 3,

"capacity": "3-person"

},

"amenities": ["Wi-Fi", "Water"],

"food": {

"available": true,

"type": "Home-cooked",

"price": "Included"

},

"price": 5000,

"photos": ["storage\_url1", "storage\_url2"],

"created\_at": "2025-05-01"

}

* **Inquiries Collection**:

json

{

"id": "inquiry789",

"listing\_id": "listing456",

"user\_id": "user789",

"message": "Is the room available?",

"created\_at": "2025-05-01"

}

**UI Mockups**

* **Owner Dashboard**: Form for posting/editing listings (dropdowns, photo upload), table of listings.
* **User Section**: Search page with dropdowns, results grid, listing details with contact button.
* **Responsive**: Use Tailwind CSS for quick, mobile-friendly styling.

**Action**: I can provide wireframes or guide Group B on Figma.

**5. Development Plan (1-Month Timeline)**

**Timeline (May 1–31, 2025)**

* **Week 1 (May 1–7)**:
  + **Developers**:
    - **You**: Set up Firebase (Authentication, Firestore, Storage), Express server, listing APIs (/api/listings). Draft proposal.
    - **Dev1**: Initialize React app, build owner form (dropdowns, photo upload).
    - **Dev2**: Set up user search page (dropdowns, results skeleton).
  + **Group A**:
    - Source location data (JSON with State/District/City/PIN).
    - Create 10 sample listings.
  + **Group B**:
    - Create UI mockups in Figma.
    - Research competitors (NoBroker, PG Finder).
  + **Milestone**: Proposal submitted, Firebase set up, UI skeletons ready.
* **Week 2 (May 8–14)**:
  + **Developers**:
    - **You**: Integrate Firebase Auth, build messaging system (inquiries or email).
    - **Dev1**: Complete owner dashboard (edit/delete, photo display).
    - **Dev2**: Finish user section (search results, listing details).
  + **Group A**: Add 10–20 more sample listings.
  + **Group B**: Draft report (intro, literature review).
  + **Milestone**: Core features (post, search, contact) functional.
* **Week 3 (May 15–21)**:
  + **Developers**:
    - **You**: Test APIs, optimize performance.
    - **Dev1**: Polish owner UI, ensure responsive design.
    - **Dev2**: Add search filters (price, room type), test user flow.
  + **Group A**: Test app (post as owner, search as user), report bugs.
  + **Group B**: Complete report (methodology, implementation).
  + **Milestone**: App fully functional, initial testing done.
* **Week 4 (May 22–31)**:
  + **Developers**:
    - **You**: Deploy app (Vercel/Firebase), fix bugs, prepare demo.
    - **Dev1/Dev2**: Assist with bug fixes, write unit tests.
  + **Group A**: Finalize slides, test demo flow.
  + **Group B**: Finalize report (results, conclusion), record demo video.
  + **You (Lead)**: Review report/slides, rehearse demo, submit deliverables.
  + **Milestone**: App deployed, report/slides submitted, demo ready.

**Action**: Confirm timeline or request a daily breakdown.

**Implementation Steps**

1. **Setup** (Week 1):
   * Install Node.js, React (npx create-react-app MessFinder), Firebase CLI.
   * Set up Firebase project (enable Authentication, Firestore, Storage).
   * Create GitHub repo, add developers.
   * Install: express, firebase, axios, tailwindcss.
2. **Back-end** (You, Week 1–2):
   * Firebase setup:

javascript

import { initializeApp } from 'firebase/app';

import { getFirestore } from 'firebase/firestore';

import { getStorage } from 'firebase/storage';

import { getAuth } from 'firebase/auth';

const firebaseConfig = { apiKey: "YOUR\_API\_KEY", */\* ... \*/* };

const app = initializeApp(firebaseConfig);

export const db = getFirestore(app);

export const storage = getStorage(app);

export const auth = getAuth(app);

* + Express APIs:

javascript

const express = require('express');

const { db } = require('./firebase');

const { collection, addDoc, query, where, getDocs } = require('firebase/firestore');

const app = express();

app.use(express.json());

app.post('/api/listings', async (req, res) => {

const listing = await addDoc(collection(db, 'listings'), req.body);

res.status(201).json({ id: listing.id });

});

app.get('/api/search', async (req, res) => {

const { state, district, city\_area, pin } = req.query;

const q = query(collection(db, 'listings'),

where('location.state', '==', state || null));

const snapshot = await getDocs(q);

res.json(snapshot.docs.map(doc => ({ id: doc.id, ...doc.data() })));

});

app.listen(3000);

1. **Front-end** (Dev1, Dev2, Week 1–3):
   * Owner form (Dev1):

jsx

import { useState } from 'react';

import { db, storage } from './firebase';

import { addDoc, collection } from 'firebase/firestore';

import { ref, uploadBytes } from 'firebase/storage';

const OwnerForm = () => {

const [form, setForm] = useState({ location: {}, room\_details: {} });

const handleSubmit = async () => {

const photoUrls = []; *// Upload to Firebase Storage*

await addDoc(collection(db, 'listings'), { ...form, photos: photoUrls });

};

return (

<form onSubmit={handleSubmit}>

<select name="location.state">

<option value="">Select State</option>

{/\* From Group A’s JSON \*/}

</select>

<input type="file" multiple />

<button type="submit">Post</button>

</form>

);

};

* + User search (Dev2):

jsx

import { useState } from 'react';

import axios from 'axios';

const SearchPage = () => {

const [filters, setFilters] = useState({ state: '', district: '' });

const [listings, setListings] = useState([]);

const handleSearch = async () => {

const { data } = await axios.get('/api/search', { params: filters });

setListings(data);

};

return (

<div>

<select onChange={e => setFilters({ ...filters, state: e.target.value })}>

<option value="">Select State</option>

</select>

<button onClick={handleSearch}>Search</button>

{listings.map(listing => (

<div key={listing.id}>

<h3>{listing.location.city\_area}</h3>

<p>Price: ₹{listing.price}</p>

</div>

))}

</div>

);

};

1. **Deployment** (You, Week 4):
   * Deploy front-end to Vercel (vercel --prod).
   * Host Express server on Render or Vercel.

**Action**: Share coding experience for tailored code or setup commands.

**6. Testing**

* **Functional**:
  + Owner: Post/edit/delete listings, upload photos.
  + User: Search with filters, view details, send inquiry.
* **Non-Functional**:
  + Performance: Search < 1s, UI loads < 2s.
  + Usability: Intuitive dropdowns (Group A tests).
* **Tools**: Postman, Jest, Chrome DevTools.
* **User Testing**: Group A tests flows; Group B validates report.

**Action**: I can provide test cases post-prototype.

**7. Documentation and Presentation**

* **Report** (10–15 pages, Group B):
  + Intro, problem, objectives.
  + Literature review (competitor analysis).
  + Methodology: Tech stack, Firebase, architecture.
  + Implementation: Screenshots, code snippets.
  + Results: Demo outcomes, feedback.
  + Conclusion: Limitations (no admin panel), future work (add admin).
* **Presentation** (10–12 slides, Group A):
  + Problem/solution (2 slides).
  + Demo (3 slides).
  + Tech stack/architecture (2 slides).
  + Challenges/results (2 slides).
  + Future scope (1 slide).
* **Demo** (You):
  + Show owner posting, user search, inquiry (< 5 min).
  + Use Group A’s sample data.
* **Viva Prep**:
  + Questions: Why Firebase? (Speed.) Why no admin? (Time.)
  + Review code and schema.

**Action**: Share report/slide requirements for outlines.

**8. Submission and Demo**

* **Polish**: Bug-free UI, fast search, clean demo.
* **Backup**: GitHub repo, demo video, PDF report.
* **Submission**: Per institute format.
* **Demo Tips**:
  + Script: “I’ll post a listing… Now, search as a user… Send an inquiry.”
  + Fallback: Video if live demo fails.

**Action**: I can provide a demo script or checklist.

**Addressing Your Context**

* **Tight Timeline**: Removing admin panel cuts ~20% of dev effort. Firebase simplifies setup; 1-month plan prioritizes MVP (post, search, contact).
* **Team Constraints**: Non-devs handle data, testing, and documentation, freeing developers. Clear roles prevent confusion.
* **Laziness/Worry**: Firebase and React are beginner-friendly; I’ll provide code/guides. Daily standups keep momentum.
* **No Admin Panel**: Listings are auto-published, simplifying logic and demo.

**Immediate Next Steps**

1. **Confirm Details**:
   * Exact dates (e.g., May 5–June 5).
   * Report/slide requirements.
   * Team member names for task table.
2. **Start Work**:
   * Create GitHub repo and Firebase project (I can guide).
   * Draft proposal (I can write by May 2 if you confirm).
   * Source location data (I’ll find JSON for Group A).
3. **Setup**:
   * Install Node.js, React, Firebase CLI (I’ll provide commands).
   * Share progress by May 7.

**Example Proposal**:

* **Title**: MessFinder: Connecting Owners and Tenants for Budget Stays
* **Problem**: Lack of student-focused room rental platforms.
* **Solution**: Web app with owner listings and user search.
* **Tech Stack**: React, Node.js, Express, Firebase.
* **Timeline**: May 2025 (4 weeks).
* **Team**: 3 developers, 6 non-devs.