# Breast Cancer Detection using Machine Learning (Wisconsin Diagnostic)

Project Abstract | Built-to-order software project | projectech.in

## 1. Introduction / Background

Planning a trip today happens across scattered notes apps, screenshots and spreadsheet rows. Timings drift between messages, costs are forgotten, and nobody knows the real total until the money is already spent. Existing consumer trip planners are either read-only inspiration boards or booking funnels; neither gives a small group a working day-by-day operations plan where every activity has a time and a cost. This project builds a web application that treats an itinerary as a structured, computable plan: days containing timed, categorized, costed activities, from which budgets, totals and packing lists are derived automatically.

## 2. Problem Statement

The core problem is fragmentation of trip information. A traveller keeps the flight booking in email, the hotel in another app, the day's plan in a chat thread and the running cost in their head. When plans change, every copy goes stale. Student-built planners typically stop at a static list of places, which does not answer the two questions that matter during a trip: what happens when, and what does it cost. The project addresses this by making the itinerary itself the single source of truth — one data model that renders the timeline, the budget and the checklist.

## 3. Objectives

1. Build a day-by-day itinerary planner where each activity carries a time, category, cost and notes. 2. Derive a live budget view (spend by category, spend by day, line items) from the same itinerary data with zero duplicate entry. 3. Provide a multi-trip dashboard with budget-vs-spend progress and trip status. 4. Generate a persistent packing checklist seeded from the itinerary. 5. Ship the whole application as a single deployable file with browser-local persistence.

## 4. Proposed Methodology

A trip is modelled as a JSON object: metadata plus an ordered array of days, each holding an array of activity objects {time, title, note, cost, category}. All views — timeline, budget, checklist — are pure render functions of this model, so an edit in one place propagates everywhere by re-render. The demo build uses vanilla JavaScript with hash-based routing (#/overview, #/itinerary, #/budget, #/checklist) and localStorage for persistence, keeping the stack dependency-free and the file deployable on any static host. Budget aggregation is done with two reduce passes over the activity arrays (group by category, group by day), drawn on Canvas bar

charts without any chart library.

## 5. System Architecture / Working

Presentation layer: four routed views — My Trips (dashboard cards with progress bars), Day-by-Day (tabbed timeline with add/delete forms), Budget (canvas charts + line-item table), Packing (persistent checklist). Data layer: a single in-memory trip object plus a localStorage-backed store for user-created trips. Logic layer: itinerary renderer, budget aggregation engine, packing seeder, and trip creator. Adding an activity inserts into the day's array, sorts by time, re-renders the timeline, and the next visit to the Budget view recomputes charts from the updated model — demonstrating the single-source design end to end.

## 6. Hardware / Software Requirements

Software: any modern evergreen web browser (Chrome, Edge, Firefox, Safari); no build step, no server, no database for the demo build. Development: a text editor and the demo HTML file (approx. 22 KB). Hardware: any laptop or desktop capable of running a modern browser. Optional extensions (full project): a Node.js/Python backend with a relational database for accounts, shareable links and PDF export.

## 7. Expected Outcomes

A working single-file web app with a complete 5-day demo itinerary (21 timed, costed activities), a live budget dashboard, a trip creator and a packing checklist. The budget view is expected to reconcile exactly with hand-computed totals from the itinerary, since both derive from the same arrays. Deliverables include the app, a project report, a presentation and a viva Q&A; document covering data modelling, state management, hash routing and the extension path to a backend.

## 8. Applications

Personal and group trip planning; college excursion and industrial-visit planning; travel-agency itinerary drafting for clients; event planning where timed schedules and budgets matter (weddings, fests); and as a teaching example of single-source data modelling in front-end development.

## 9. Future Scope

User accounts with cloud sync; shareable read-only trip links for group members; PDF itinerary export; map integration with route distances and travel-time estimates between activities; currency conversion for international trips; collaborative editing with conflict resolution; and a native-feel PWA with offline support.

## 10. Conclusion

The Travel Itinerary Planner demonstrates that a well-modelled domain object removes whole categories of app complexity: one itinerary model drives the timeline, the budget and the checklist. The project is scoped honestly — browser-local in the demo build, with a documented path to a full-stack product — and gives the student a complete, demonstrable product plus strong viva material on data modelling and front-end architecture.

Keywords: breast cancer detection, WDBC, random forest, medical machine learning, classification, final year project