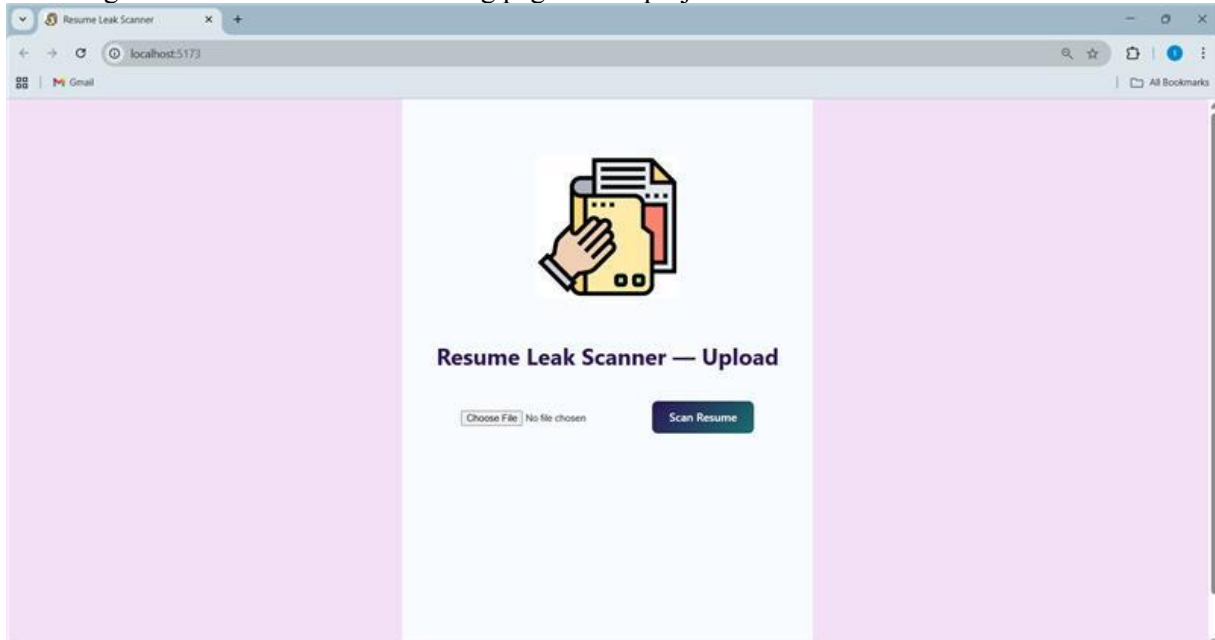
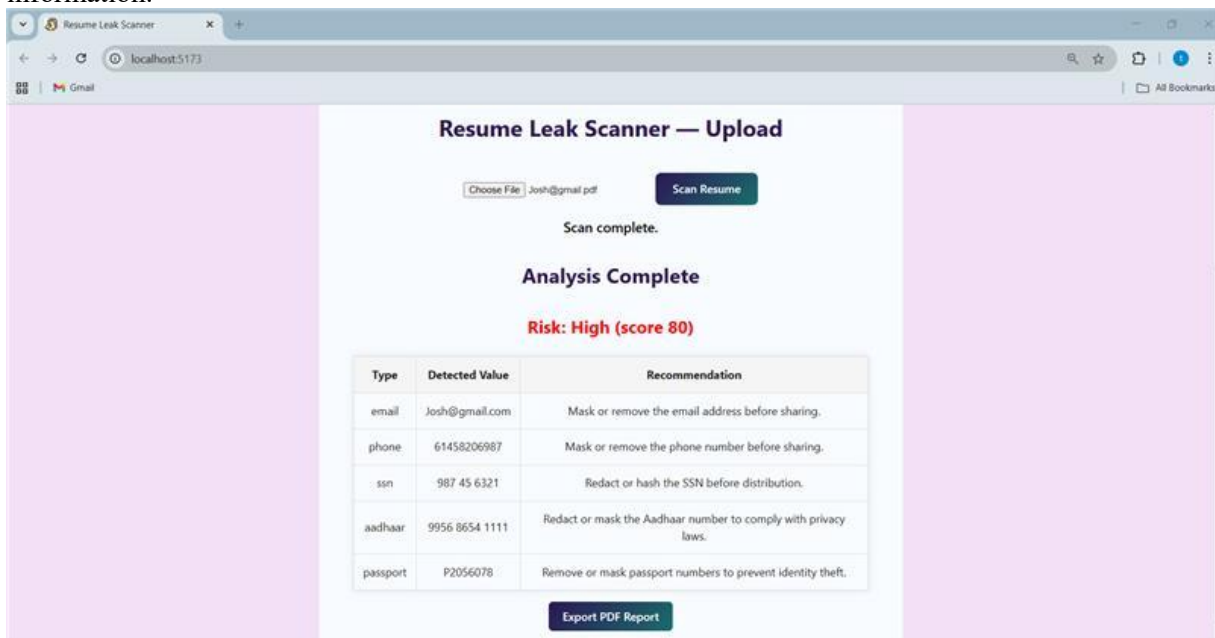


A series of screenshots of the application in action:

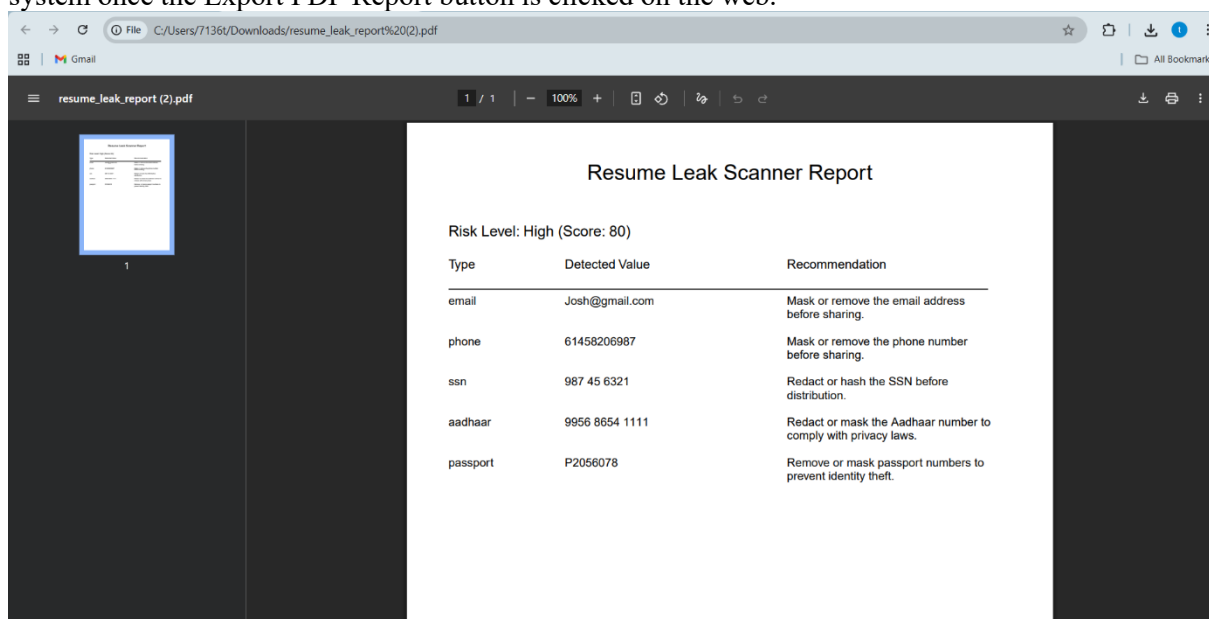
The image below shows the main landing page of our project.



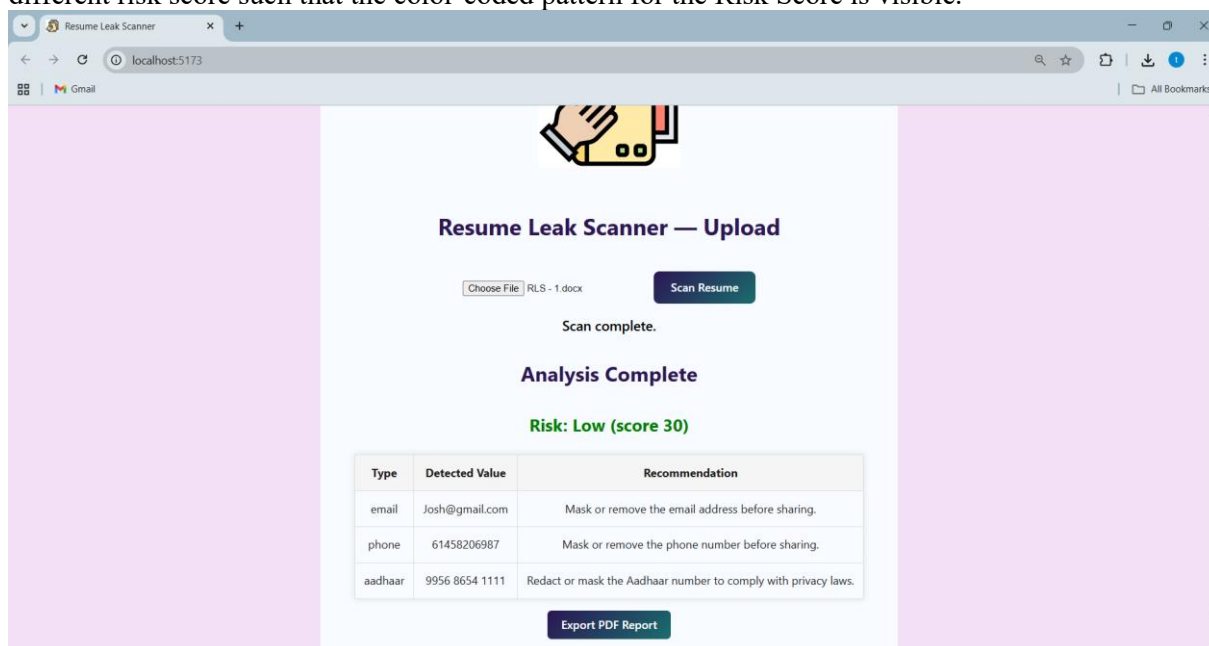
The image depicts the output screen once the sample document is uploaded and the Scan Resume button is clicked after that. The outcome is of the table consisting of the type of sensitive data, the detected respective value in the file and the possible avoidable recommendation for each of the information.

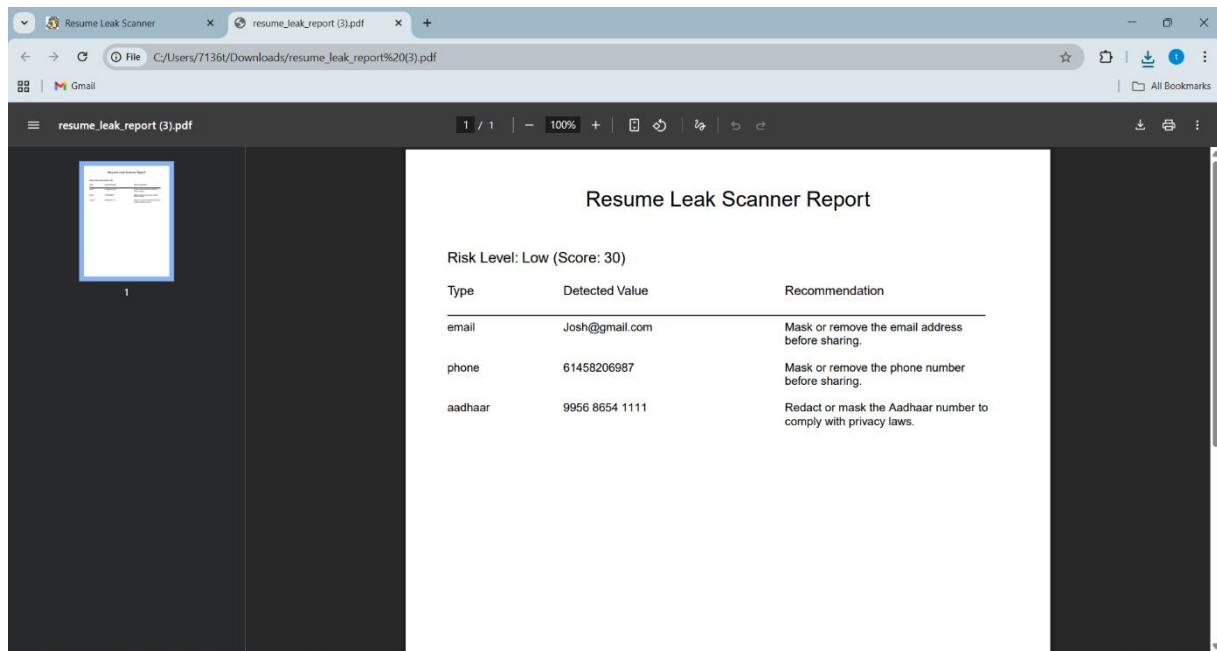


The image below is the outcome of the downloaded PDF file that gets downloaded into the personal system once the Export PDF Report button is clicked on the web.

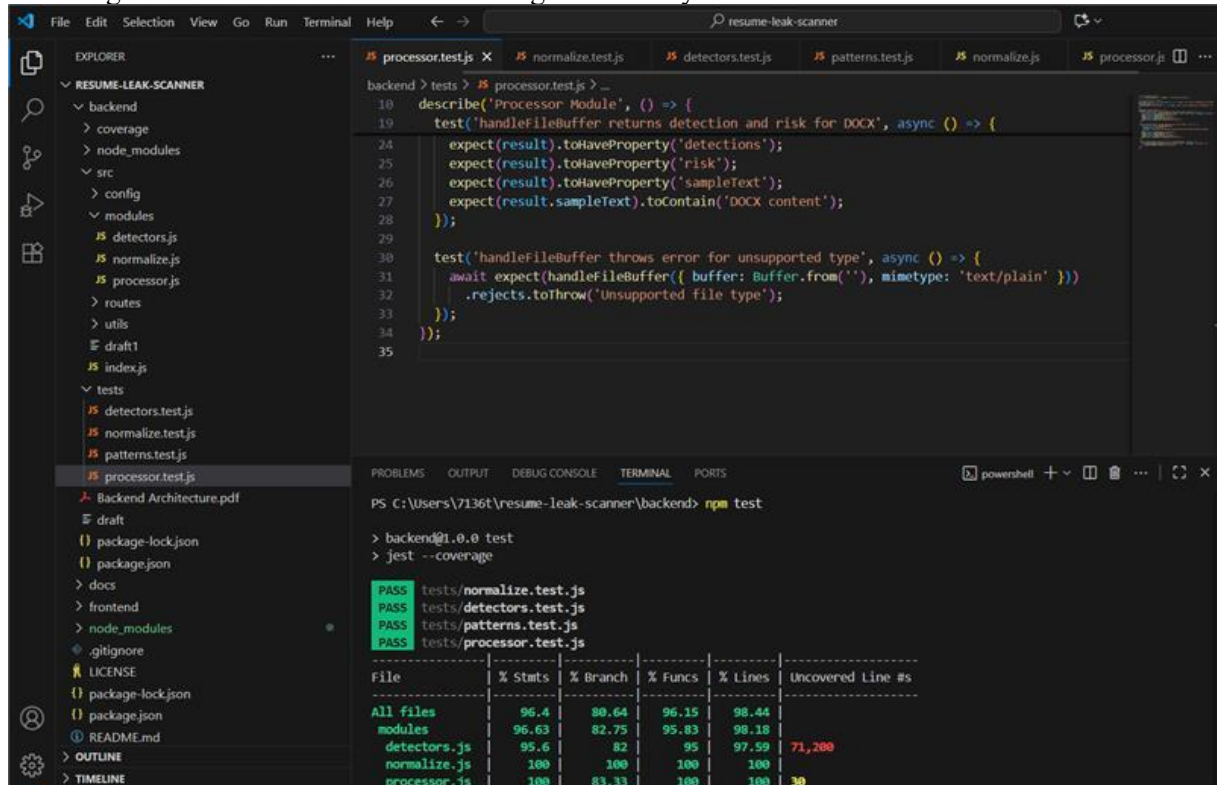


The below two images are just another outcome illustrations for a different file consisting of a different risk score such that the color-coded pattern for the Risk Score is visible.





The images below show the testcases running successfully for both the text extraction and detection.



The screenshot displays the Visual Studio Code interface for the 'resume-leak-scanner' project. The Explorer sidebar on the left shows the project's file structure, including folders for 'backend', 'coverage', 'node_modules', 'src', 'config', 'modules', 'routes', 'utils', 'draft1', 'index.js', and 'tests'. The 'tests' folder is expanded, showing several test files: 'detectors.test.js', 'normalize.test.js', 'patterns.test.js', and 'processor.test.js'. The 'processor.test.js' file is currently open in the editor, showing Jest test cases for the 'Processor' module. The bottom panel shows the 'Test Results' view, which includes a table of file coverage and a summary of test results.


File	96.4	80.64	96.15	98.44
All files	96.4	80.64	96.15	98.44
modules	96.63	82.75	95.83	98.18
detectors.js	95.6	82	95	97.59
normalize.js	100	100	100	100
processor.js	100	83.33	100	100
utils	95	50	100	100
idDetector.js	92.85	50	100	100
patterns.js	100	100	100	100




Test Suites: 4 passed, 4 total
 Tests: 12 passed, 12 total
 Snapshots: 0 total
 Time: 3.726 s
 Run all test suites.
 PS C:\Users\71361\resume-leak-scanner\backend>




Section 4: Individual contributions



Name	Main Role(s)	Target Grade
Tejasri Kanneganti	Core developer (backend parsing + frontend integration), Scrum Master Sprint 1, Documentation lead	Credit
Joshitha Yalamanchili	Backend lead (Express server, detection engine), Scrum Master Sprint 2	Distinction
Shruthi Bhaskaran	Backend Developer, Data Extraction Engineer, Frontend Developer-UI Designer	Pass





Tejasri Kanneganti

Criteria	Pass -> "can do allocated work with guidance"	Credit -> "performs their role in the Project"	Distinction -> "can see themselves in the team and assist others"	High Distinction -> "can demonstrate excellence"
Core Development		 <p>Implemented multiple backend features to the Resume Leak Scanner, such as regex patterns for ID and email detection, integration of the pdf-parse library for text extraction, and the</p>		

		establishment of a backend module to calculate risk levels based on identified entities. We expanded functionality by adding a new export endpoint for PDF reports, ensuring that both the detection and reporting processes operated seamlessly.		
CRUD				
Design		 <p>Completed and configured the Resume Leak Scanner's backend folder structure and overall repository, making sure that modules, routes, and tests are clearly separated. arranged the backend such that the team could quickly integrate and maintain export functionality, risk scoring, and detection modules, creating a clear framework for future growth.</p>		
SRS		 <p>Made a substantial contribution to the SRS paper by outlining functional requirements linked to detection, such as risk rating, PDF text extraction, and regex-based email and ID identification. helped create a strong specification by making ensuring non-functional requirements pertaining to precision, effectiveness, and scalability were also precisely stated.</p>		
Testing		 <p>Created Jest unit tests for the detection, normalization, and text extraction modules, confirming the correctness of regex</p>		

		matches for sensitive data, the dependability of normalized outputs, and the proper parsing of resumes in DOCX and PDF formats. recorded anticipated inputs and outputs, enhancing trust in the accuracy of the system.		
Real Time				
Discuss with tutor during workshops				
Leadership				
Tools	Pass	Credit	Distinction	High Distinction
GIT		 <p>Set up the repository structure for the Resume Leak Scanner project and consistently commit code updates, including new modules and tests, to ensure clean version control. Used branches for development, then merged tested contributions into the main branch for team use.</p>		
Trello		 <p>Personal tasks were tracked with clear labels, including regex development, backend modules, export endpoints, and testing progress. Statuses were updated on a frequent basis to enable the team keep track of development progress and dependencies.</p>		
README file		 <p>Improved the README file by adding setup instructions, clarifying the project structure, and presenting example API usage, making the documentation easier to understand for anyone running the Resume Leak Scanner locally.</p>		

Criteria	Pass -> "can do allocated work with guidance"	Credit -> "performs their role in the Project"	Distinction -> "can see themselves in the team and assist others"	High Distinction -> "can demonstrate excellence"
Core Development	 <p>Integrated pdf-parse in Processor.js to extract and normalize text from PDFs for detecting emails, phone numbers, and addresses. On the frontend, the results page will display the risk score dynamically after upload, provide a "Download PDF" option linked to the backend export endpoint, and feature an improved, responsive design for enhanced readability. Integrated all detection rules into the backend to ensure resumes could be scanned and processed effectively.</p>			
CRUD				
Design				
SRS	 <p>Drafted the external interface requirements for the Resume Leak Scanner, detailing how users interact with the system and how the system communicates with external components. Compiled and integrated these requirements into the final SRS document, ensuring a complete and coherent specification for the</p>			

	team.			
Testing	<div></div> <p>Tested your own code for the Resume Leak Scanner by creating and verifying regex patterns for address detection. Ensured that the frontend risk score integration, the “Download PDF” button, and the improved user interface for the results page are all working correctly by testing with sample PDFs. Recorded testing progress and results in Trello cards.</p>			
Real Time				
Discuss with tutor during workshops				
Leadership				
Tools	Pass	Credit	Distinction	High Distinction
GIT	<div></div> <p>Pushed all personal code changes to the team Git repository, ensuring the latest version of the Resume Leak Scanner was up-to-date.</p>			
Trello	<div></div> <p>Used Trello to track assigned tasks, moving items across stages (To Do → In Progress → Done) to reflect progress in the project workflow.</p>			
README file	<div></div> <p>Documented basic project setup and instructions for running the Resume Leak Scanner locally,</p>			

	including dependencies and initial configuration steps.	
--	---	--

Joshitha Yalamanchili

Criterion	Self-Assessment	Evidence	Evidence
Core Development	Distinction	<ul style="list-style-type: none"> -Helmet, CORS, dotenv, and /health route are all included in this bootstrapped Express server. -Multer memoryStorage was used to implement the /api/upload endpoint (5 MB limit, MIME checking). -Created a processor module to parse PDF (pdf-parse) and DOCX (mammoth) buffers. -Normalized text, integrated detection engine (regex for addresses, ZIP codes, phones, and emails), and risk score calculation. -JSON output containing detections, risk, and sample text is provided by connected backend modules. 	Github_s223681984_R esume Leak Scanner
Design	Distinction	<ul style="list-style-type: none"> -Bootstrap was integrated for a responsive front-end user interface. -Color-coded risk levels were used in the frontend suggestions table design. -The ability to export PDFs was added. -A backend architecture diagram that was co-authored and illustrates the connections between upload, parsing, detection, and storage. 	Backend Architecture
SRS	Distinction	<ul style="list-style-type: none"> -Outlined the functional and non-functional needs (speed, error-handling, scalability) for regex detection and backend file handling. -Created a privacy and security checklist for managing sensitive data. -DOCX/PDF parsing, detection, and recommendation flow documentation. 	SRS Doc
Testing	Distinction	-Used sample PDF/DOCX files	Github Link for

		<p>with addresses, phone numbers, and email addresses to do manual end-to-end checks.</p> <p>-For the detection engine (emails, phones, IDs), Jest unit tests were implemented in backend/tests/detectors.test.js.</p> <p>-For automated testing, a npm test script was included.</p> <p>-Frontend display and PDF export features were tested.</p>	Testings
Leadership	Distinction	<p>-As a Scrum Master, I managed sprint planning and led retrospectives and stand-ups.</p> <p>-Offered technical advice on module integration, code standards, and backend architecture.</p> <p>-Helped colleagues with debugging, regex implementation, and Git process.</p> <p>-Made sure that the team's integration and development procedures were uniform.</p> <p>-I created a team contribution document that maps tickets to individual members and records time allocations to track duties and progress.</p>	Trello Board Link Github Repository Link
Tools			
GIT	Distinction	<p>-To ensure consistent integration, all module development was done on a single development branch.</p> <p>-One last pull request was made for code review and main branch merging.</p>	Github Repository Link

		<ul style="list-style-type: none"> -Maintained consistent commit messages and a clean commit history. -Coordinated module integration and helped colleagues with Git workflow. 	
Trello	Distinction	<ul style="list-style-type: none"> -Each task and contribution had a thorough Trello card created. -Contains comments, labels, descriptions, and checklists. -Updated card statuses to monitor dependencies and progress. -Assisted teammates in comprehending and adhering to the card format. 	Trello Board Link
README	Distinction	<ul style="list-style-type: none"> -Risk scoring, detection modules, PDF/DOCX handling, frontend configuration, and backend endpoints were all documented. -Presented rules for testing, usage examples, and setup instructions. -Contains instructions for data validation, role-based access control, and JWT/OAuth authentication. -Made sure the README is understandable and available to future users and colleagues. 	GIT README